UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Estudio e implementación de técnicas de alineamiento de grandes volúmenes de voz y texto

Máster Universitario en Investigación e Innovación en Inteligencia Computacional y Sistemas Interactivos

Autor: Luis Miguel Martínez Antolín
Tutor: Diego de Benito Gorrón
Ponente: Doroteo Torre Toledano

JUNIO 2020

Trabajo Fin de Máster

ESTUDIO DE TÉCNICAS DE ALINEAMIENTO DE GRANDES VOLÚMENES DE VOZ Y TEXTO

AUTOR: Luis Miguel Martínez Antolín
DIRECTOR: Diego de Benito Gorrón
PONENTE: Doroteo Torre Toledano

AUDIAS
Dpto. de Tecnología electrónica y de las comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
JUNIO 2020

Resumen

Resumen

Los sistemas de reconocimiento automático del habla son aquellos sistemas encargados de decodificar la señal de voz obteniendo como salida una transcripción en forma de texto. Este tipo de tecnología está avanzando a gran velocidad en la actualidad, estando presente en la mayoría de las grandes empresas, a pesar de algunas limitaciones como la ausencia de grandes volúmenes de datos de libre acceso en español, necesarios de cara a su aprendizaje y entrenamiento. Para construir estas bases de datos es necesaria la información de alineamiento entre los datos de audio y voz que la componen.

En este Trabajo de Fin de Máster se han analizado las principales técnicas de alineamiento de voz y texto de uso libre en la actualidad, con el fin de determinar cuál de ellas es más adecuada de cara a alinear grandes bases de datos con las que entrenar sistemas de reconocimiento de voz. Se realizó una investigación acerca de la mayoría de las herramientas de alineamiento forzado de la actualidad, valorando cada una de ellas y escogiendo dos para realizar una evaluación de cara al problema que se plantea. Después, se implementó un alineador forzado de forma original a partir de un sistema de reconocimiento de voz del grupo AUDIAS, y se evaluó de la misma forma que los alineadores anteriores, comparando los resultados entre sí. En el segundo Trabajo de Fin de Máster, mostrado en [1] se continuará con el proceso, utilizando el alineador que se considere óptimo en la elaboración de una base de datos en español de acceso libre.

Palabras Clave

alineamiento forzado, alineador, audio, texto, voz, reconocimiento del habla, bases de datos

Abstract

Automatic Speech Recognition is a technology that decodes a voice signal and outputs it in text form. This type of technology is currently advancing at great speed, despite some limitations such as the absence of large volumes of freely accessible data, necessary for learning and training. To build these databases, the alignment information between the audio and voice data that make up the database is necessary.

In this final Master's Thesis, the main techniques for aligning voice and text that are currently used freely have been analysed in order to determine which of them is more appropriate for aligning large databases with a view to training voice recognition systems. An investigation was carried out on most of the current forced alignment tools, evaluating each one and choosing two to carry out an evaluation in view of the problem that arises. Then, a forced aligner was implemented in an original way from a speech recognition system of the AUDIAS group, and it was evaluated in the same way as the previous aligners, comparing the results with each other. In the following Master's Thesis, shown in [1], the process will be continued, using the aligner that is considered optimal in the development of a freely licensed Spanish database.

Keywords

forced alignment, aligner, audio, text, voice, speech recognition, speech corpus, asr

Agradecimientos

A mi tutor y mi ponente, Diego y Doroteo, por ayudarme en todo momento y por todo lo que he aprendido con ellos durante la realización de este trabajo.

A todo el grupo de AUDIAS, y en especial a Beltrán, por estar siempre que lo he necesitado disponible para echarme una mano, y a Alicia por introducirme en este mundo.

A mis compañeros del máster.

Índice general

Índice de figuras

Indice de tablas

Capítulo 1

Introducción

1.1. Motivación del proyecto

Los sistemas de reconocimiento automático del habla o *Automatic Speech Recognition* (ASR) son tecnologías que permiten transformar una señal de voz en su transcripción de texto. Este tipo de sistemas está en auge en los últimos años, teniendo gran relevancia en infinidad de tecnologías, como los teléfonos móviles o la gestión de documentos (búsquedas por voz). A la hora de implementar un sistema de reconocimiento automático del habla serán necesarias dos informaciones fundamentales. La primera de ellas, será la correspondiente a la información relacionada con la señal de voz que se reconocerá, y se encontrará en el llamado Modelo Acústico. La segunda de ellas, será la información sobre el vocabulario utilizado en el tipo de habla, y estará caracterizada en el Modelo de Lenguaje. Para poder generar de forma fiable los dos modelos mencionados es necesario tener disponible una base de datos que contenga los ficheros de audio a reconocer y las transcripciones de texto correspondientes, así como los datos de alineamiento temporal entre cada uno de los ficheros de audio y texto. Una de las principales dificultades a la hora de implementar un sistema de ASR es la ausencia de grandes bases de datos de libre acceso que permitan el entrenamiento de estos sistemas. Además de conseguir grandes cantidades de horas de audio con sus transcripciones, es necesario tener la información acerca de la correspondencia temporal (alineamiento) entre cada audio y su fichero de texto correspondiente, de forma que si los ficheros de audio son muy grandes se puedan segmentar de forma precisa. Esta información es difícil de conseguir y complica la tarea de construir este tipo de bases de datos.

Por tanto, es interesante conocer y evaluar el rendimiento de los principales sistemas de alineamiento temporal de audio y texto, de cara a poder construir estas bases de datos de forma fiable, y sin la necesidad de realizar los alineamientos manualmente, lo que llevaría un excesivo número de horas de trabajo.

El principal objetivo de este trabajo será el de evaluar algunos de los alineadores forzados de audio-texto más eficaces de la actualidad y evaluarlos sobre una base de datos que se obtendrá de forma manual, con la finalidad de verificar cúal de ellos funciona mejor. También se desarrollará la implementación de un sistema de alineamiento original, que se comparará con los anteriores. El sistema que sea más preciso será de gran ayuda de cara a construir una base de datos de gran tamaño y licencia libre, tarea que se desarrollará en el Trabajo de Fin de Máster complementario.

1.2. Objetivos y enfoque

El objetivo principal de este trabajo es la evaluación de diferentes alineadores de audio y texto, así como la implementación de uno nuevo de cara a poder alinear grandes cantidades de ficheros de audio y texto. De este modo se podrán elaborar bases de datos que sirvan para el entrenamiento de nuevos sistemas de reconocimiento de voz. Los objetivos parciales referentes al primero de los trabajos de fin de máster son los siguientes:

Documentación y elección de los distintos alineadores de audio y texto mas efectivos. Instalación de los alineadores a evaluar.

Obtención manual de los datos de audio y texto para la evaluación y alineamientos manuales de los ficheros de audio y texto, que se tomarán como referencia para la evaluación.

Procesamiento de las medidas de error entre los datos de referencia y los obtenidos por los alineadores, con el objetivo de ver a qué se deben las diferencias entre alineamientos.

Reconocimiento de los ficheros de audio mediante un reconocedor de voz adaptado a al problema planteado, para obtener su transcripción, de cara a implementar el nuevo alineador, así como alineamiento entre las transcripciones de texto extraídas del reconocedor y los ficheros originales de texto.

Evaluación del nuevo alineador de audio y texto y comparación con el rendimiento de los alineadores evaluados anteriormente.

1.3. Organización de la memoria

- **Capítulo 1. Introducción**

  Este capítulo contextualiza el trabajo en el marco de los sistemas de reconocimiento de habla, la creación de bases de datos y los sistemas de alineamiento de audio y texto, que serán la referencia principal a la hora de desarrollar este Trabajo de Fin de Máster.

- **Capítulo 2. Estado del arte**

  En este capítulo se trata sobre todos los conceptos previos necesarios para la comprensión de los sistemas de alineamiento de audio y texto, así como se explica el funcionamiento de aquellos sistemas que se van a evaluar, de cara a determinar cuál es el más preciso.

- **Capítulo 3. Descripción y diseño de los experimentos**

  Se exponen las diferentes pruebas que se harán para los diferentes alineadores de audio y texto, así como de los métodos que se emplearán para implementar el alineador original. Además, se detallan los pasos a seguir de cara a obtener unos datos de referencia que permitirán realizar la evaluación de forma fiable.

- **Capítulo 4. Resultados**

  Los resultados finales acerca de la evaluación de los alineadores de audio y texto se muestran en este capítulo, así como las diferentes medidas y representaciones gráficas que se obtienen a partir de los datos obtenidos durante los distintos experimentos.

- **Capítulo 5. Conclusiones y trabajo futuro**

  Tras observar los resultados obtenidos, este capítulo resumirá las conclusiones obtenidas durante todo el trabajo, así como las posibles líneas de trabajo futuro dentro de esta área.

Capítulo 2

Estado del arte

2.1. Los sistemas de reconocimiento de voz

El proceso de reconocimiento automático del habla es un proceso de decodificación y transcripción de la señal de voz. Los sistemas de ASR reciben como entrada una señal de audio procedente de un hablante y capturada a partir de un micrófono, y se encargan de analizarla a partir de determinados patrones, modelos o algoritmos, dando como resultado de salida la información de la señal de voz en forma de texto. [2]

Estos sistemas requieren dos principales modelos que permiten asociar probabilidades para cada una de las frases y palabras que aparecen en la señal de voz, con la finalidad de obtener aquellas que más probabilidad tienen de haberse dicho, consiguiendo así la transcripción final en formato de texto.

El primero de los modelos es el **Modelo Acústico**. Este se encarga de predecir los fonemas que más probabilidad tienen de haber sido pronunciados para cada segmento de la señal de voz.

El segundo modelo, correspondiente al **Modelo de Lenguaje**, es el encargado de predecir qué frases son más probables de pronunciarse dadas las palabras reconocidas anteriormente. Por tanto, ayuda a dar información contextual y permite construir aquellas frases que tengan más sentido, ya que si no se tendría una sucesión de palabras reconocidas, válidas morfológicamente, pero no sintácticamente. Para construir el modelo de lenguaje es necesario un texto de entrenamiento así como el **Léxico**, que contiene la información fonética de cada una de las palabras que se pronuncian en el lenguaje a reconocer. Es decir, en él se encuentra la información de pronunciación de cada una de las palabras conocidas por el reconocedor.

Para generar estos modelos se utilizan principalmente los Modelos Ocultos de Markov o *Hidden Markov Models* (**HMMs**), aunque en los últimos años se han estado sustituyendo progresivamente por el uso de sistemas basados en redes neuronales recurrentes o *Recurrent Neuronal Networks* (**RNNs**). En el próximo punto se tratará sobre estos modelos más detalladamente. [3]

2.2. Dynamic Time Warping

El *Dynamic Time Warping* o **DTW** es una técnica que sirve para encontrar un alineamiento óptimo entre dos señales temporales. Este algoritmo se implementó originariamente para utilizarse en sistemas de ASR. El objetivo principal de DTW es comparar dos secuencias dependientes del tiempo, de igual o diferente longitud, determinadas bajo una serie de restricciones.

1. Cada índice de la primera secuencia debe estar alineado con uno o más índices de la segunda secuencia.
2. El primer índice de la primera secuencia tiene que estar alineado con el primero de la segunda.
3. El último índice de la primera secuencia tiene que estar alineado con el último índice de la segunda secuencia.
4. El alineamiento de los índices de la primera secuencia con la segunda tiene que ser monótono creciente, de modo que no haya cruces entre alineamientos en ningún punto. Es decir, si un índice **i** de la primera secuencia se alinea con un índice **k** de la segunda, nunca un índice $\mathbf{n} > \mathbf{k}$ de la segunda secuencia podría alinearse con un índice $\mathbf{l} < \mathbf{i}$ de la primera.

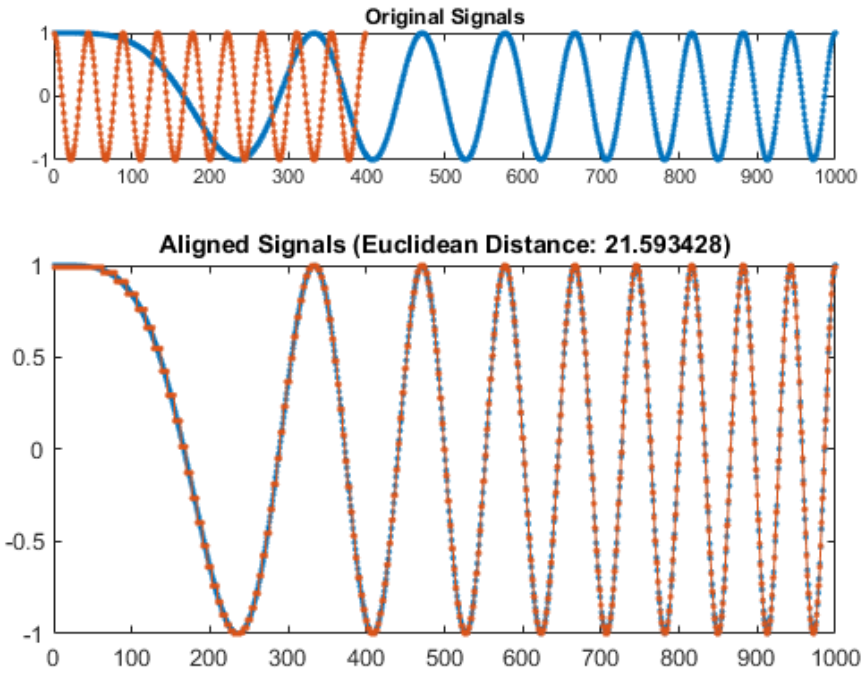

Figura 2.1: Alineamiento entre dos señales (DTW)

El alineamiento óptimo entre las dos señales será aquel que, cumpliendo dichas restricciones, tenga una función de coste menor, siendo la función de coste del algoritmo el sumatorio de todas las diferencias absolutas entre los valores de cada par de puntos alineados.

Cada punto de la señal no tiene por qué estar linealmente unido al de su índice correspondiente en la señal a comparar. Para calcular las distancias entre todos los puntos, así como la

secuencia de alineamientos entre ellos, se calcula una matriz de distancias que permite ver con claridad el alineamiento entre los puntos de cada señal.

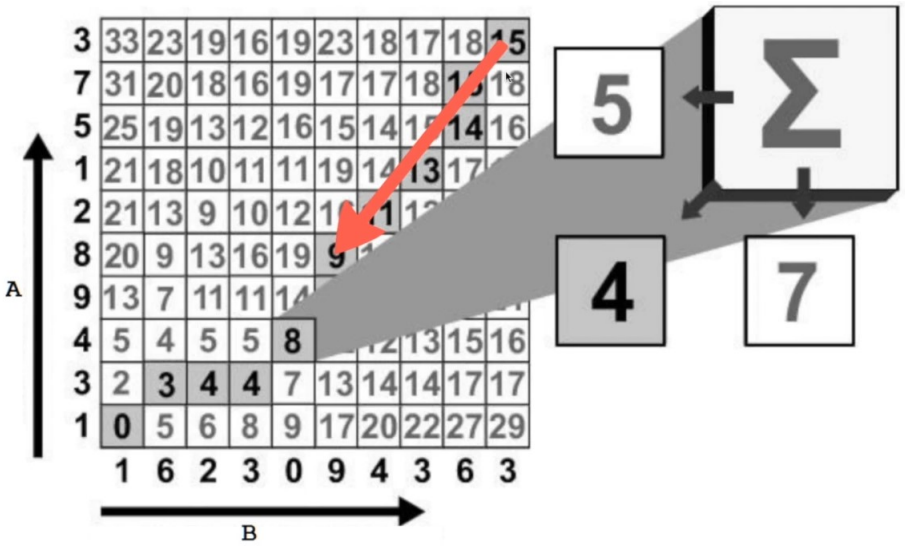

Figura 2.2: Matriz de distancias de DTW

En la figura 2.2 se puede ver la matriz de distancias calculada entre dos señales. Para calcularla, se parte de la distancia euclídea entre los dos puntos a comparar, a la que se le sumará adicionalmente la distancia mínima calculada entre las tres comparaciones que se hayan hecho previamente. De este modo, no sólo se tiene en cuenta la distancia entre cada par de puntos, sino que se tiene en cuenta de la semejanza de los anteriores cercanos. Este método permite comparar dos señales de longitud diferente, sin necesidad de tener que comparar cada índice de cada señal con el correspondiente de la otra señal. [4]

2.3. Modelos Ocultos de Markov (HMMs)

Los Modelos Ocultos de Markov o *Hidden Markov Models* (**HMMs**) han sido hasta hace poco tiempo los modelos más utilizados en reconocimiento del habla. Se encargan de modelar de forma estadística la acústica de la señal de voz, es decir, cómo suenan los diferentes fonemas y palabras. Estos modelos están siendo sustituidos progresivamente por las redes neuronales, aunque también se pueden utilizar en modelos híbridos que combinan ambos sistemas.

El objetivo es predecir parámetros de la señal de voz desconocidos a partir de otros conocidos observables, para lo que se utiliza un diagrama de estados que se define mediante el número de estados, la matriz de probabilidades de transición entre estados, el número de parámetros observables y las probabilidades iniciales de cada estado. De este modo, se toman como parámetros o estados del diagrama los diferentes fonemas de la señal de voz, con el objetivo de predecir aquellos fonemas cuya probabilidad de haber sido pronunciados haya sido mayor.

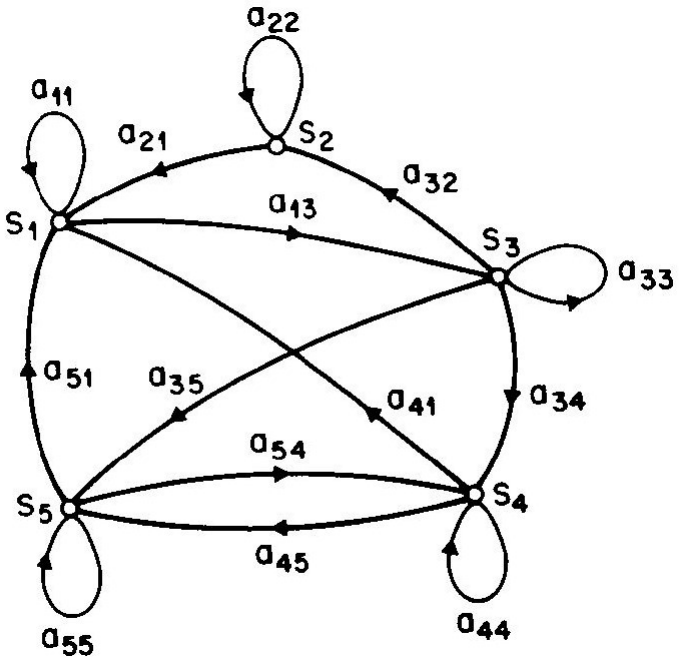

Figura 2.3: Diagrama de estados de un HMM.

El algoritmo más utilizado para estimar los parámetros de los HMMs, que maximiza la probabilidad de la observación dado el modelo es el algoritmo *Baum-Welch*, un caso particular del algoritmo *Expectation-Maximization* [5] aplicado a los **HMMs**.

El algoritmo utilizado para ver qué secuencia de estados (fonemas) es más probable que suceda a partir de unos parámetros observables es el algoritmo de *Viterbi*. [6]

2.4. HTK: Herramienta de Modelos Ocultos de Markov

HTK es un conjunto de herramientas de software de código abierto desarrolladas por la Universidad de Cambridge en 1994 para construir y manejar Modelos Ocultos de Markov. Consiste en una serie de módulos de librerías y un conjunto de más de 20 herramientas (programas). Está implementada en **ANSI C** y comenzó funcionando principalmente en sistemas **UNIX**, aunque en la actualidad puede ejecutarse en cualquier sistema operativo moderno. Se utiliza principalmente en sistemas de reconocimiento de voz y análisis de la señal de audio, basándose en modelos HMM.

La librería de HTK contiene diez módulos que proveen una interfaz entre las herramientas a utilizar y el usuario, así como una variedad de funciones de apoyo de gran utilidad. Los módulos de dicha librería son los mostrados en la siguiente tabla. [7]

Nombre	**Módulo**
HDbase	Base de datos de tokens de entrenamiento
HGraf	Control gráfico de señales de audio
HLabel	Control de etiquetas de entrada y salida
HMem	Administración de memoria
HModel	Interpreta las definiciones de HMM de entrada/salida
HParse	Soporte gramatical
HShell	Interfaz del sistema operativo
HSigP	Rutinas de procesamiento de señal
HSpIO	Control de datos de voz de entrada y salida

Tabla 2.1: Modulos de la librería de HTK

Estos módulos se utilizan mediante una herramienta de entrenamiento que lee una serie de recetas de HMM y unos datos de entrenamiento asociados, produciendo nuevos archivos de HMM. En los últimos años se ha ampliado el número de estos módulos, entre los que destacan los siguientes ejemplos. **HAudio**, encargado de controlar la captura de señales de audio. **HMath**, que se encarga de operar con estructuras de alto nivel que sirven para manejar los datos con mayor facilidad (vectores, matrices, etc). Por último, **HML** es un modelo dedicado únicamente a los modelos de lenguaje.

Nombre	**Función**
HAlign	Realiza alineamientos forzados
HCode	Análisis de voz (LPC,MFCC, etc)
HDEd	Editor de diccionario en modo batch
HERest	Re-estimación de Baum-Welch integrada
HHEd	Editor de HMM en modo batch
HInit	Unidad asilada segmentada de K-means para la inicialización del modelo
HLEd	Editor de archivos de etiquetas en modo batch
HList	Lista de contenidos de un fichero de datos
HRest	Re-estimación Baum-Welch aislada
HResults	Análisis de resultados
HSLab	Editor de ficheros simple e interactivo
HSource	Generación de datos utilizando una fuente estadística de HMM
HVite	Decodificador Viterbi (Aislado y conectado)

Tabla 2.2: Principales funciones de HTK

A parte de estos módulos, **HTK** se construyó con más de 25 herramientas de partida que sirven para construir sistemas basados en HMMs, y se utilizan unas u otras dependiendo del tipo de HMMs requeridos. En la tabla 2.2 se pueden ver las principales herramientas con las que cuenta HTK.

A partir de estas funciones se realizan tareas como la parametrización de los datos (**HCode**), o el entrenamiento de sistemas de **ASR** con gran facilidad (**HERest**).

A partir de los diferentes módulos y herramientas definidas, **HTK** se ha utilizado para una cantidad muy grande de aplicaciones y sistemas. En 1994 su licencia se utilizaba en más de 100 laboratorios del habla en todo el mundo. Originalmente su uso se limitaba al desarrollo de sistemas ASR, sin embargo en la actualidad se utiliza en cualquier sistema que pueda modelarse como un HMM. Ha resultado ser una herramienta de gran rendimiento en sistemas de reconocimiento y síntesis de voz, pero también en otras áreas como el etiquetado automático de texto y audio para infinidad de bases de datos, o incluso campos no relacionados con el procesamiento de la señal de voz, como el Proyecto Genoma (secuenciación del ADN) o análisis de imagen y reconocimiento de patrones. [8]

En cuanto a su licencia es algo limitado, ya que permite su uso libre para investigación y enseñanza, pero establece una serie de restricciones para su uso comercial y distribución, en comparación con otras herramientas más modernas, como **Kaldi**, que se verá a continuación.

2.5. Kaldi: Herramienta para sistemas de ASR

Kaldi es una herramienta de código abierto utilizada principalmente para sistemas de ASR, implementada en **C++** y bajo la licencia **Apache v2.0**, una de las menos restrictivas dentro de las disponibles. Aparte de su licencia, es más moderno, flexible y tiene un mejor soporte que sus competidores, HTK y RASR (RWTH ASR) [9].

Su estructura se conforma de la siguiente manera. En primer lugar, para la infraestructura se utiliza la librería **OpenFST**. Como librerías para el álgebra numérico se utilizan las librerías **BLAS** y **LAPACK**. En función de cuál de las dos clases de librería se utilicen se conformarán dos subgrupos de librerías. A cada una de las funcionalidades de estas herramientas se accede a través de una terminal de comandos creada en **C++**. Cada una de estas herramientas tiene funciones específicas de reconocimiento de voz. Por ejemplo, hay ejecutables diferentes para acumular estadísticas, sumar acumulaciones de éstas, y, por último, generar un modelo acústico mediante GMMs utilizando las estimaciones de máxima verosimilitud. En la tabla de la figura 2.4 se puede ver la jerarquía de librerías con las que cuenta Kaldi, desde los comandos de la terminal o scripts hasta las librerías internas que utiliza para los diferentes procesos. [10]

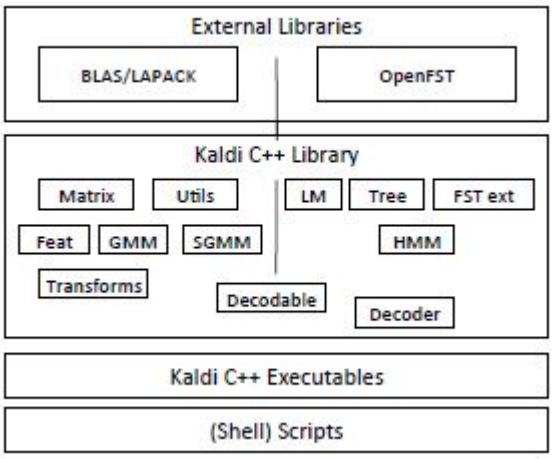

Figura 2.4: Jerarquía de librerías de Kaldi.

Dentro de las opciones que tiene para extraer características de la señal de voz, hay una gran variedad, como por ejemplo *Mel Frequency Cepstral Coefficients* (**MFFC**), *Cepstral Mean and Variance Normalization* (**CMVN**), etc. Además, se tiene libertad para modificar las diferentes opciones de cada uno de estos coeficientes, como por ejemplo, el número de bins de los **MFCCs**.

En cuanto al modelado y entrenamiento de sistemas ASR, ofrece gran variedad de posibilidades, desde el uso de GMMs/HMMs hasta redes neuronales profundas o recurrentes (**DNN**, **RNN**). Estas dos últimas han sido añadidas recientemente, ya que Kaldi es una herramienta en constante desarrollo, y permite ir actualizándose a medida que nacen nuevos modelos para entrenamiento de sistemas de reconocimiento de voz. [10]

2.6. Sistemas de alineamiento de audio y texto

Un alineador forzado es un sistema que permite realizar alineamientos entre el habla y su transcripción ortográfica, obteniendo como salida un mapa de sincronización que asocia cada intervalo de texto a un intervalo de tiempo (dentro del archivo de audio).

Por tanto, los sistemas de alineamiento de audio-texto permiten encontrar un nexo entre la información de la señal de voz y su transcripción. Dentro de su uso se encuentran los sistemas de indexado multimedia (para búsquedas de voz o texto), así como sistemas de síntesis de voz, y, finalmente, sistemas ASR, que serán el objetivo para el que se realizará la evaluación.

Dentro de los alineadores se pueden distinguir dos principales tipos. El primero de ellos está basado en el reconocimiento de la señal de voz, mediante un sistema ASR, para después alinear la transcripción obtenida con el texto original. Este alineamiento consiste en obtener qué fonemas se parecen más a cada parte de la señal de audio, a su forma de onda y espectrograma, y dependerán de los modelos acústicos, construidos, en la mayoría de casos, mediante el uso de HMMs, implementados en Kaldi o HTK. Estos modelos acústicos pueden estar pre-entrenados (con bases de datos alineadas previamente e incorporadas al alineador) o entrenados mediante un proceso de entrenamiento/alineamiento simultáneo en el que los datos de entrenamiento serán los mismos para los que se generarán las etiquetas de alineamiento. [11]

En el segundo tipo, mucho menos común, se utiliza un sintetizador de voz para convertir el texto en una señal de audio, que se comparará con la señal de voz original para así obtener una correspondencia temporal entre ambas.

Además, dentro de los alineadores forzados de la actualidad, la gran mayoría requiere una serie de condiciones para obtener un alto rendimiento. Lo primero y más importante, es necesario que se acompañe la señal de voz junto a su transcripción de texto. Muchos de ellos son capaces de corregir errores en las transcripciones incorrectas, para alinear de nuevo con las correcciones aplicadas. También es importante que posibles errores durante el alineamiento no afecten a alineamientos de etiquetas temporales posteriores. En caso de haber errores en la transcripción, muchos de los modelos no los corregirán pero serán capaces de realizar alineamientos con las partes que correspondan. Todos deben tener una capacidad de memoria suficiente para guardar todas las posibles ponderaciones sobre qué etiquetas generar, sobre todo en modelos probabilísticos como HMMs o GMMs. Por último, en la mayoría de ellos consta de un sistema de ASR interno, que permite alinear audio y texto para idiomas para los que no se tienen modelos acústicos y de lenguaje entrenados. [12]

Dentro de las licencias de los alineadores disponibles, la mayoría de ellos están protegidos comercialmente, con precios muy elevados, que van desde los 1000 euros anuales hasta los 10.000 euros por la licencia del programa completo. Debido a ello, es importante conocer el rendimiento de los pocos alineadores gratuitos a los que se pueden acceder a día de hoy.

A continuación se describirán algunas de las principales herramientas de alineamiento de libre licencia de la actualidad, para probar aquellas que sean más acordes al problema que se plantea, y realizar una evaluación viendo cuál ofrece mejores resultados. Posteriormente se implementará una herramienta propia con el objetivo de mejorar los resultados de las evaluadas previamente.

2.7. Alineadores de uso libre en la actualidad

El número de alineadores de audio-texto disponibles en la actualidad es muy pequeño. Si a esto se le añade que la mayoría requiere el pago de una licencia de precio muy elevado, dificulta aún más la investigación en este tipo de sistemas.

Nombre	**Algoritmo**	**Idiomas**	**Interfaz**	**Código**	**Licencia**
Aeneas	DTW	30+	CLI, LIB, Web	Python, C	AGPL
CMUSphinx	HMM(Propio), RNN	11	CLI, LIB	Java, C, Python	Similar a MIT
Kaldi	HMM(Propio), DNN, RNN	Inglés	CLI, LIB	C++	Apache
Montreal Forced Aligner	HMM (Kaldi)	Inglés	CLI	Python	MIT
SailAlign	HMM(HTK)	Inglés	CLI	Perl	GPL
FAVE	HMM(HTK)	Inglés	CLI	Perl	GPL
MAUS	HMM(HTK)	21	CLI, Web	C	All Rigths Reserved
DARLA	HMM(HTK)	Inglés	Web	Perl	GPL
Gentle	HMM(Kaldi)	Inglés	CLI,Web	Python	MIT
P2FA	HMM(HTK)	Inglés	CLI, Web	Pyhton	Propia
ProsodyLab	HMM(HTK)	Inglés, Griego, Español	CLI	Perl	GPL
SPPAS	HMM(Julius)	5+	GUI, CLI	Python	GPL
Julius	HMM(Propio)	Inglés, Japonés	CLI, LIB	C	Similar a MIT
LaBB-CAT	HMM(HTK)	Inglés	CLI, Web	Java	GPL
EasyAlign	HMM(HTK)	4+	CLI	C, C++ (Praat plugin)	Propia
PraatAlign	HMM (HTK)	Universal	CLI	Python, C, C++ (Praat plugin)	Propia
FaseAlign	HMM(HTK)	Español	CLI, Web	Python	Propia
SPRAAK	HMM(Propio)	Italiano	CLI, API Propia	C, Python	All Rights Reserved

Tabla 2.3: Diferentes alineadores de audio-texto en la actualidad

En cuanto a los distintos tipos de alineadores de uso libre que se tienen disponibles, se realizó una investigación exhaustiva sobre todos los modelos en los que su código estaba disponible, para ver cuáles pueden ser más interesantes de cara a utilizarlos en la generación de las etiquetas necesarias para construir una base de datos.

En la tabla 2.3 se pueden ver sus características principales, entre las que destaca lo siguiente. En primer lugar, es importante que, dentro de los idiomas soportados, se encuentre el español, ya que será el utilizado para construir la base de datos. En segundo lugar, en cuanto a la licencia de cada uno, todos permiten su descarga y uso de forma libre, pero no su comercialización o uso para construir otras herramientas. Dentro de las licencias más permisivas, se encuentra la

licencia Apache, MIT y AGPL/GPL. A continuación se tratará sobre algunos de los aspectos más importantes de cada uno de ellos de cara a su posible utilización en este TFM.

En primer lugar, **CMUSphinx**, al ser un sistema de ASR, su instalación y utilización como alineador, por separado, resulta muy compleja, además de no asegurar buenos resultados para el español, puesto que está entrenado para utilizarse en inglés. [12]

El alineador forzado de Kaldi [10] es uno de los más avanzados, utilizando una arquitectura propia de HMMs y DNNs/RNNs, pero la dificultad para utilizar modelos de distinto idioma al inglés complican su utilización para los datos de prueba.

En cuanto a **MAUS**, es un software que se encuentra disponible en el pack de señales de audio bávaro (Schiel, 1999). Puede instalarse tanto en el ordenador del usuario como utilizarse vía Web a través de **WebMAUS**, y tiene modelos pre-entrenados para diferentes idiomas, como el inglés y el alemán, así como otros menos usuales, como los aborígenes de Islandia y Australia, por ejemplo. En las evaluaciones que se realizaron para este alineador, se vio superado en rendimiento por alineadores como el **Montreal Forced Aligner**, el **FAVE** o **LaBB-CAT**, lo que sumado a su licencia, muy restrictiva, hace que no sea uno de los elegidos para este trabajo. [11].

Uno de los alineadores que, en lugar de utilizar HTK, utilizan Kaldi para generar sus modelos, es **Gentle**. Es una herramienta interesante debido a su licencia de Kaldi, poco restrictiva. Sin embargo, su interfaz Web no funciona, y su instalación es muy compleja, por lo que no se evaluará. [13].

DARLA es una herramienta de extracción de información lingüística que hace uso a su vez de otras herramientas para realizar la mayoría de sus tareas. Para los alineamientos forzados utiliza **FAVE** y **MFA**, por lo que no se entrará en profundidad en su rendimiento. [14]

El **P2FA** (Penn Phonetics Lab Forced Aligner) es una herramienta de alineamiento forzado basada en HTK. Se caracteriza por su sencillo uso en un script de Python y contiene modelos pre-entrenados de inglés americano. Su última actualización consta de 2009, por lo que hay otros mucho más avanzados en la actualidad. [15]

El siguiente en la tabla 2.3 es el sistema de código abierto **Julius**, que llama la atención ya que está entrenado para funcionar en japonés y se desconoce su precisión en otros idiomas. [16]

Para continuar la investigación se eligió el **Montreal Forced Aligner** o Alineador Forzado de Montreal (MFA), ya que a pesar de no incluir el español como idioma para su uso, tiene una gran capacidad de generalización y escalabilidad a otros idiomas, a diferencia de otros similares, como **FAVE**. En el siguiente punto se entrará en profundidad en las características de este modelo, ya que será uno de los cuales se utilizará en los experimentos a realizar. [17]

El **ProsodyLab Aligner** podría tener sentido en la evaluación, ya que tiene pre-entrenados modelos acústicos y de lenguaje para el español. Sin embargo, el **Montreal Forced Aligner** está implementado partiendo de este modelo como base, y obtiene mejores resultados, y es por ello por lo que se utilizará en la evaluación de los alineadores. Además, su licencia es mucho más restrictiva que la del **MFA** por utilizar HTK en lugar de Kaldi en su modelo de HMMs, tal y como se explica en [17].

Por último, el alineador **Aeneas**, soporta más de 30 idiomas, su licencia es la más permisiva, y, a pesar de que el algoritmo en el que se basa es tan simple como es el caso del DTW, ofrece muy buenos resultados. Además, no se ha hecho una evaluación que aporte datos sobre el rendimiento de esta herramienta, por lo que es interesante incluirlo en la evaluación. En el siguiente punto se entrará en profundidad para comprender su funcionamiento.

A continuación se tratará con más detalle en algunas herramientas que, a pesar de no haber sido elegidas para este trabajo, son interesantes de cara a líneas futuras de evaluación.

2.7.1. SailAlign

SailAlign es un alineador forzado para segmentos de larga duración de gran robustez, y de gran aceptación en la comunidad científica. El algoritmo que utiliza para alinear, basado en HTK, se desarrolla de la siguiente forma.

Inicialización Se divide el audio en segmentos más pequeños. Para no dividir una palabra por la mitad, se emplea un módulo de VAD (Voice Activity Detection) que permite saber en qué instantes se encuentra la señal de voz y en cuáles hay un silencio. También se realiza la extracción de características de la señal de audio.

Reconocimiento de voz. Alineamiento texto-texto Se aplica un módulo de reconocimiento de voz para identificar el contenido léxico de cada segmento. Esta parte será similar a la que se utilizará en el alineador propio, basado en un reconocedor de voz mediante HMMs. Posteriormente, las hipotéticas transcripciones se concatenan en una sola, y se alinean con la transcripción real o de referencia. Los segmentos que no se alineen debidamente se repartirán en segmentos de duración determinada, y se repetirá el proceso de alineamiento para estos.

Adaptación del modelo acústico y de lenguaje Para mejorar la robustez frente al ruido, se adaptan los modelos acústicos en cada iteración de forma supervisada utilizando los alineamientos de confianza. Los modelos de lenguaje también se actualizan, entrenándose específicamente para aquellas regiones en las que no se haya conseguido el alineamiento.

SailAlign obtuvo buenos resultados para la base de datos de inglés **TIMIT**, pero su antigüedad (2011) y su falta de modelos para el español lo hacen poco apto para utilizarlo en la construcción de nuestra base de datos. [18]

2.7.2. FaseAlign, EasyAlign & PraatAlign

Fase Align [19] es un alineador forzado pre-entrenado mediante HMMs utilizando la herramienta HTK. Los modelos acústicos, así como diccionario, se han entrenado para funcionar para una variedad de dialectos de español latino muy grande. A pesar de que la licencia de HTK es más restrictiva que la de otras herramientas como Kaldi, y que los alineamientos se realizarán para español castellano, se harán algunas pruebas con él, debido a la facilidad que tiene para ejecutarse y a estar preparado para funcionar sin necesidad de un entrenamiento previo. [19]

El alineador forzado **EasyAlign** [20] está desarrollado sobre el software de alineamiento manual **Praat** y está basado en modelos HMM de HTK. El proceso de alineamiento que usa sigue los siguientes pasos. Se realiza una segmentación de la transcripción de texto de entrada, en párrafos, líneas, o por signos de puntuación. En segundo lugar se realiza una conversión de nivel ortográfico a nivel fonético para estas transcripciones (G2P). Por último, se realizan las predicciones sobre alineamientos entre fonemas mediante HMM. Los resultados que ofrece son buenos para la evaluación mediante segmentos de voz en inglés y francés, con un error menor de 20ms en un 80 % de las etiquetas. Además, cuenta con un sistema para añadir nuevos diccionarios de distintos idiomas con facilidad, y se añadieron modelos acústicos y diccionario de español poco antes de la publicación del trabajo, por lo que es una opción viable para evaluarlo en el problema que se plantea.

Otro alineador forzado similar, basado también en **Praat**, es **PraatAlign**. Este se diferencia en que utiliza modelos acústicos entrenados en **MAUS**, y tiene una capacidad de generalización muy grande, por lo que también sería apto para hacer pruebas con él, pero la tendencia debido a su similitud es que los resultados sean similares a los obtenidos en **EasyAlign**. [21]

2.7.3. LaBB-CAT

El siguiente alineador es el **LaBB-CAT Forced Aligner**. Se trata de una herramienta de gestión de bases de datos creada, inicialmente, para investigación de carácter socio-fonológico. Se diseñó para gestionar cantidades muy grandes de grabaciones de habla, transcripciones ortográficas y anotaciones lingüísticas. Además de estas funciones, adquirió la de alinear segmentos de audio y texto, utilizando HMMs en **HTK** mediante un sistema de entrenamiento/alineamiento simultáneo, que consta de los siguientes pasos.

1. Mediante una capa de HTK de transcripción fonética se asocia cada identificador de palabra con su pronunciación correspondiente.
2. Se extraen los fonemas y transcripciones ortográficas y se genera un diccionario de pronunciación a partir de la capa de transcripción fonética (G2P). Esto, normalmente, se hará para cada hablante.
3. Se entrenan los modelos acústicos y de lenguaje monofonema, utilizando los datos generados en el paso anterior.
4. Se alinean los datos utilizando los modelos entrenados previamente.

Además de este proceso, en caso de alineamientos para segmentos de menos de cinco minutos de voz, utiliza modelos pre-entrenados, por no ser suficientes los datos de entrada, que serían a su vez los de entrenamiento. Además, contiene diversas funcionalidades relacionadas con la extracción de características no lingüísticas (por ejemplo, sociales) y con el etiquetado de datos. [22]

2.7.4. SPRAAK & SPPAS

El siguiente alineador dentro de nuestra investigación es el **SPRAAK**. Se trata de, una vez más, un alineador que se encuentra dentro de una herramienta general de reconocimiento de habla, como es el caso de algunos de los alineadores anteriores. Se encontró esta herramienta debido a sus buena acogida para implementar sistemas de ASR en italiano. Sin embargo, su licencia permite su uso libre, pero es muy restrictiva, y es por ello por lo que se ha descartado. [23]

Similar al anterior alineador se encuentra **SPPAS**. Es una herramienta diseñada para lingüistas y que permite realizar un análisis automático de la métrica o prosodia del habla. Su uso básico es el de segmentar la señal de voz a nivel de fonema, sílaba o palabra, así como su transcripción. Esta utiliza **Julius** como herramienta de alineamiento, utilizando modelos entrenados para el inglés mediante HTK. La licencia de HTK tiene restricciones para la distribución de sistemas que utilicen sus funcionalidades, sin embargo esto no ocurre con los modelos entrenados a partir de sus herramientas, y es por ello que **SPPAS** cuenta con una licencia del tipo *General Public License* (GPL). Puesto que es imposible realizar pruebas con todos los alineadores encontrados, explorar las posibilidades de este modelo para los datos de prueba entra dentro de las lineas de trabajo futuro. [24]

2.7.5. Alineador Forzado de Montreal (Montreal Forced Aligner)

El alineador forzado con mejor rendimiento en la actualidad a nivel de palabra y fonema es el Alineador Forzado de Montreal (**MFA**). Sin embargo, se desconoce cómo será el rendimiento de esta herramienta, diseñada específicamente para segmentos de duración corta, en los segmentos utilizados en la base de datos de la evaluación, que serán de larga duración. Además, el MFA está diseñado para funcionar en inglés, y sus pruebas se realizaron en inglés, por lo que es interesante ver hasta dónde llega su capacidad de obtener un buen rendimiento para idiomas como el español.

Esta herramienta se caracteriza principalmente por tres características que lo mejoran frente a sus antecesores. En primer lugar, está implementado en **Kaldi**, cuya licencia es más permisiva que otras herramientas como **HTK**. La principal ventaja de este alineador pasa por la mejora de su arquitectura. El MFA incorpora una arquitectura mejorada más compleja, que utiliza modelos de trifonemas con adaptación al locutor, en lugar de modelos de monofonemas sin adaptación. Otra mejora respecto a los sistemas anteriores es la escalabilidad a la hora de entrenar el modelo con una variabilidad de datos muy grande, lo que lo hace exportable a otras bases de datos diferentes, con distintos idiomas y condiciones acústicas. En cuanto al rendimiento que ofrece, se midieron diferencias entre etiquetas manuales y alineamientos automáticos realizados por el MFA, a nivel de palabra y fonema, para tres bases de datos diferentes, obteniendo los siguientes resultados. Un 2-5 % de las etiquetas tienen diferencias de al menos 100 ms, mientras que un 90 % tiene diferencias de menos de 50 ms. Estos resultados mejoran los de los alineadores forzados anteriores, como **PLA** (*Prosody-Lab Aligner*) y **FAVE**, y por ello será por lo que se incluirá en los alineadores de la evaluación. [17]

2.7.6. Aeneas Alignment Tool

Dentro de las herramientas de licencia libre de la actualidad para realizar alineamientos entre audio y texto se encuentra el alineador Aeneas. Este modelo fue implementado en 2015 con una gran acogida, ya que ofrecía muy buenos resultados a pesar de su licencia no restrictiva. Previamente a la creación de Aeneas, se encontraban disponibles algunos modelos de alineadores gratuitos que habían desarrollado grupos de investigación, todos ellos con alguna de las siguientes limitaciones.

En primer lugar, todos carecían de modelos acústicos y de lenguaje pre-entrenados correspondientes a idiomas que no fueran el inglés.

En segundo lugar, su instalación y ejecución era demasiado compleja para que la mayoría de los usuarios fueran capaces de utilizarlo con facilidad. Además, si el usuario no tiene conocimientos relacionados con el procesamiento de lenguaje, su uso se dificulta aún más.

Por último, la licencia de muchos de ellos no permitía utilizarlos con fines comerciales.

La principales ventajas de Aeneas de cara a al objetivo final del trabajo, que tal y como se explica en 1.1, trata de la construcción de una base de datos de licencia libre en español, son las siguientes.

Tiene un alto rendimiento en idiomas diferentes al inglés, como por ejemplo, el español en este caso. Los buenos resultados de Aeneas en general, y en concreto en español, se conocen dado que a simple vista los alineamientos que realiza son precisos, pero no se ha realizado ninguna evaluación de forma rigurosa a partir de etiquetas de Aeneas y etiquetas manuales fiables, por lo que es interesante realizar una evaluación que permita ver su rendimiento real.

Aeneas no utiliza herramientas complejas de ASR como **Kaldi** o **HTK**, sino que está basado en una estructura mucho más sencilla. En primer lugar, utiliza un sistema de **TTS (*Text-to-***

***speech*)** para transformar el texto en audio sintético. Este audio no necesita ser de gran calidad, sino que siendo inteligible es suficiente para realizar la tarea de alineamiento que se realiza a posteriori. Una vez se cuenta con el audio original y el sintético, se aplica el algoritmo **DTW** (*Dynamic Time Warping*) visto en 2.2 a los **MFCCs** de cada una de las señales, obteniendo así las mejores correspondencias entre las dos señales de audio. Dado que el audio sintetizado va asociado a cada una de las partes del texto, se tendrá un alineamiento entre el audio original y dicho texto asociado.

En cuanto al código, se desarrolla en **Python**, lo que supone una mayor libertad a la hora de manipular ficheros de configuración, escribir y leer ficheros de texto, etc. Dentro de Python se utilizan programas como **eSpeak** para la síntesis del audio o **FFMpeg** para la conversión de formatos de los ficheros para el procesamiento de la señal, así como módulos que permiten recoger las llamadas a estos programas y recoger los resultados. Además, su instalación, aunque compleja en algunos sistemas operativos, es muy sencilla en Windows o Mac, con un ejecutable 'todo en uno' que permite instalarlo con facilidad.

Por último, se distribuye con la licencia **AGPL** (*Affero General Public License*) que ofrece las siguientes ventajas. Permite descargar el código fuente, utilizarlo, así como modificarlo de forma gratuita. También permite su uso con fines comerciales, incluyendo su venta tras aplicar modificaciones o vender servicios basados en él, siempre que no se modifique, y se referencie a la licencia original, de cara a contribuir a la comunidad de Aeneas. En la descripción de los experimentos realizados se explicará la evaluación realizada en Aeneas para datos de etiquetas fiables en español. [25]

Capítulo 3

Descripción y diseño de los experimentos

3.1. Obtención de los datos de referencia

Para realizar una evaluación de un alineador forzado es necesario tener una base de datos revisada manualmente, de forma que se tenga la certeza de que puede tomarse como base de referencia y que sus datos no contienen ningún error.

La base de datos con la que se realizarán las pruebas consta, en primer lugar, de 20 ficheros de audio correspondientes a fragmentos de audiolibros. Estos fueron obtenidos de la web **LibriVox** [26]. Esta página cuenta con una cantidad muy grande de audiolibros en diferentes idiomas, leídos por una gran variedad de voluntarios y bajo la licencia de libre uso y distribución **Creative Commons** (CC). Los ficheros de audio se descargaron manualmente, y se editaron para que todos tuvieran una duración en torno a 300 segundos. Para cortar los audios se implementó un ejecutable en **Python** utilizando la función **AudioSegment** de la librería **PyDub**.

También fue necesario contar con la transcripción exacta de cada uno de estos ficheros, para lo que se utilizó como referencia la web **Project Gutenberg** [27], una página similar a LibriVox, pero contando con archivos de texto en lugar de almacenar ficheros de audio.

El siguiente paso fue realizar un alineamiento manual entre cada uno de los ficheros de audio y sus correspondientes transcripciones, de forma que se obtuviera unos valores de tiempo entre los cuales se encuentran cada uno de los segmentos de texto de cada fragmento. Para generar estas etiquetas se utilizó el Software para edición de audio y estudio de fonética acústica **WaveSurfer**, para lo que se tomó como marca de silencio el punto medio de cada intervalo de silencio entre frases, mostrado en la figura 3.1.

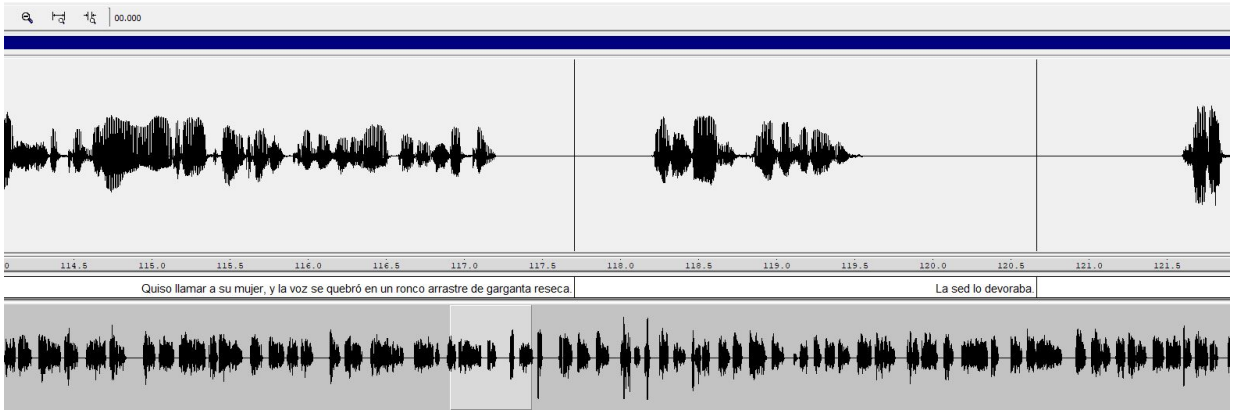

Figura 3.1: Alineamiento manual visualizado en WaveSurfer

El alineamiento se hizo a nivel de frase, tomando como divisores entre frases cada uno de los puntos de cada texto. Esto se debe a que que el objetivo final es utilizar el alineador para crear una base de datos y ficheros de audio muy grandes, y se gastaría una cantidad excesiva de tiempo en crear las etiquetas a nivel de palabra y fonema. No es necesario que el alineador tenga un gran rendimiento a nivel de palabra o fonema, sino a nivel de frase, ya que esta es la información requerida para entrenar reconocedores de voz, que es la aplicación principal del TFM. En total se etiquetaron 554 frases de duraciones que oscilan entre los 2 segundos y los 25 segundos, y llevó entre unas 15-20 horas realizar este proceso.

Una vez se realizó el etiquetado de todos los alineamientos en todos los ficheros de la base de datos, WaveSurfer genera archivos automáticos de extensión **LAB** que contienen la información de qué segmento de cada texto se encuentra entre qué instantes temporales del fichero de audio, todo ello en formato de texto, tal y como se muestra en la figura 3.2. Estos ficheros se procesarán en **Matlab** para poder ver las diferencias con aquellas etiquetas generadas por los distintos alineadores.

```
0.0000000 8.1081912 Como todas las tardes, la barca-correo anuncio su llegada al Palmar
    con varios toques de bocina.
8.1081912 26.0341922 El barquero, un hombrecillo enjuto, con una oreja amputada, iba de
    puerta en puerta recibiendo encargos para Valencia, y al llegar a los espacios
    abiertos en la unica calle del pueblo, soplaba de nuevo en la bocina para avisar su
    presencia a las barracas desparramadas en el borde del canal.
26.0341922 31.1980403 Una nube de chicuelos casi desnudos seguia al barquero con cierta
    admiracion.
31.1980403 47.0095130 Les infundia respeto el hombre que cruzaba la Albufera cuatro
    veces al dia, llevandose a Valencia la mejor pesca del lago y trayendo de alla los
    mil objetos de una ciudad misteriosa y fantastica para aquellos chiquitines criados
    en una isla de canas y barro.
47.0095130 57.9120808 De la taberna de Canamel, que era el primer establecimiento del
    Palmar, salia un grupo de segadores con el saco al hombro en busca de la barca para
    regresar a sus tierras.
57.9120808 68.0698148 Afluian las mujeres al canal, semejante a una calle de Venecia,
    con las margenes cubiertas de barracas y viveros donde los pescadores guardaban las
    anguilas.
```

Figura 3.2: Alineamiento manual de referencia en formato LAB

3.2. Experimentos mediante Aeneas y MFA

En esta sección se hará una descripción sobre las pruebas y experimentos que se realizaron para determinar el rendimiento de tres alineadores forzados: **MFA**, **Aeneas**, y, por último, el implementado de forma original, cuyo proceso de implementación se explica en 3.3. Es conveniente realizar una evaluación de cada uno de los alineadores por separado, ya que cada uno de ellos actúa de una forma diferente.

3.2.1. Alineamiento y procesado de datos de Aeneas

Para realizar los alineamientos con **Aeneas** únicamente es necesario pasarle como entrada cada par de ficheros de audio y texto, y especificar la carpeta de salida.

Tras la ejecución, se obtuvo un fichero de texto en formato **JSON**, en el que se encuentra cada frase del texto asociada a un intervalo de tiempo en el que se pronunció dicha frase, como se puede ver en la figura 3.3. Se obtuvo un intervalo por cada frase, y cada frase está determinada por cada punto y salto de línea, por lo que se pudo forzar al alineador para alinear aquellos fragmentos de texto que más interesaron, en función de las etiquetas de referencia. De este modo, se pudieron hacer comparaciones entre alineamientos que corresponden a los mismos fragmentos de audio y texto, para tener así una evaluación más precisa.


```
{
 "fragments": [
  {
   "begin": "1.840",
   "children": [],
   "end": "8.920",
   "id": "f000001",
   "language": "spa",
   "lines": [
    "Como todas las tardes, la barca-correo anuncio su llegada al Palmar con varios
        toques de bocina."
   ]
  {
   "begin": "8.920",
   "children": [],
   "end": "26.560",
   "id": "f000002",
   "language": "spa",
   "lines": [
    "El barquero, un hombrecillo enjuto, con una oreja amputada, iba de puerta en puerta
         recibiendo encargos para Valencia, y al llegar a los espacios abiertos en la
        unica calle del pueblo, soplaba de nuevo en la bocina para avisar su presencia a
         las barracas desparramadas en el borde del canal."
   ]
  },
```


Figura 3.3: Alineamiento de Aeneas en formato JSON

Para procesar los ficheros **JSON** se implementó una herramienta en **Matlab** que consta de las siguientes funciones. La primera se llama **LabFilesEval.m**, y permite almacenar todos los intervalos de tiempo entre los que se encuentran cada una de las etiquetas manuales definidas en 3.2, generadas en **WaveSurfer** en la figura 3.1. La siguiente función es **AeneasFiles.m**, en la que se leen los valores de los tiempos de las etiquetas de la misma forma en la que se hace en **LabFilesEval.m**. Antes de leer cada intervalo se verifica que dentro de cada uno se encuentra un fragmento de texto, ya que en ocasiones se crean intervalos vacíos correspondientes a silencios, y estos no se generaron en las etiquetas manuales de extensión LAB.

3.2.2. Alineamiento y procesamiento de datos de MFA

Para ejecutar el **MFA** se realizaron varios pasos. El primero consistió en, a partir de un modelo de transcripción fonética G2P (*Grapheme to Phoneme*) obtener el diccionario de pronunciación de los 20 textos a evaluar. Después, a partir de este diccionario y de cada uno de los ficheros de audio y texto, se entrenó modelo acústico y de lenguaje del alineador para cada fichero, y posteriormente el alineamiento entre cada par de ficheros.

Esto dio como salida archivos de texto de extensión **TextGrid** en los que se encuentran los alineamientos entre cada frase, palabra y fonema y el tiempo en segundos en el que se dice cada uno en el audio.

Para visualizar los archivos de texto en formato **TextGrid** se utilizó el software de análisis de audio **Praat**, como se puede ver en la figura 3.4.

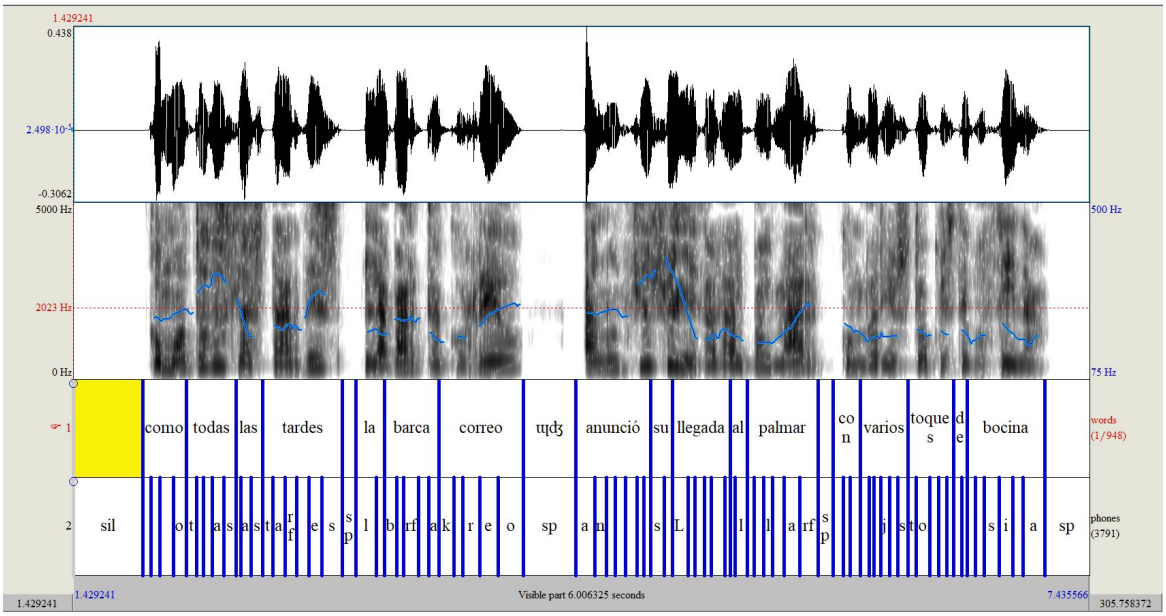

Figura 3.4: Alineamiento del MFA visualizado en Praat

Después de comprobar manualmente que los alineamientos son correctos, se procesaron los archivos de extensión TextGrid para obtener los alineamientos a nivel de frase, ya que en la salida del alineador se obtienen los alineamientos a nivel de palabra y fonema.

Para ello se hizo otro script de Matlab en el que, a partir de las frases alineadas y los tiempos de inicio y final de cada una en Aeneas y los alineamientos de palabra del MFA, se adquirieron los tiempos de inicio y final de los segmentos alineados en Aeneas. Previamente, se tuvieron que cargar todos los datos de palabras y tiempos de los ficheros TextGrid en Matlab, para lo que se usaron funciones del paquete de herramientas **mPraat** [28].

```
File type = "ooTextFile"
Object class = "TextGrid"
xmin = 0.0
xmax = 313.1939375
tiers? <exists>
size = 2
item []:
        item [1]:
                class = "IntervalTier"
                name = "words"
                xmin = 0.0
                xmax = 313.1939375
                intervals: size = 973
                        intervals [1]:
                                xmin = 0.0
                                xmax = 1.840
                                text = ""
                        intervals [2]:
                                xmin = 1.840
                                xmax = 2.110
                                text = "como"
                        intervals [3]:
                                xmin = 2.110
                                xmax = 2.390
                                text = "todas"
                        intervals [4]:
                                xmin = 2.390
                                xmax = 2.550
                                text = "las"
                        intervals [5]:
                                xmin = 2.550
                                xmax = 3.030
                                text = "tardes"
```

Figura 3.5: Alineamiento del MFA en formato TextGrid

3.3. Implementación del alineador propio

Para completar la evaluación se ha implementado un alineador forzado propio de la siguiente forma.

Se parte de un proyecto del grupo **AUDIAS** de 2016 en el que se desarrolló un reconocedor de voz basado en modelos acústicos obtenidos mediante HMMs y un tipo de *Gaussian Mixture Models* (SGMMs). Este sistema de reconocimiento de voz tiene la particularidad de que puede adaptar automáticamente el léxico y el modelo de lenguaje a un texto dado antes de reconocer un audio. De este modo, se usa el texto del audiolibro para adaptar el reconocedor al texto a reconocer, sin forzar explícitamente el texto como se haría en un alineamiento forzado. A partir de la base de datos generada en 3.1 y el reconocedor se obtienen las transcripciones de cada fichero de audio, así como un fichero de salida por cada obra, que permite obtener los instantes en los que se ha pronunciado cada palabra.

El objetivo fue el de, a partir de las transcripciones obtenidas y los instantes de pronunciación, reconstruir las etiquetas temporales equivalentes a las de referencia para hacer una evaluación similar a la de los alineadores anteriores.

Para ello se implementó en Matlab un script similar al utilizado para reconstruir los alineamientos de frase del MFA, de cara a obtener los alineamientos de frase de la transcripción del reconocedor. Sin embargo, las etiquetas obtenidas fueron muy pocas, ya que las palabras reconocidas eran parecidas, pero no idénticas a las de referencia para la mayoría de casos. También se probaron herramientas de alineamiento de textos en Python, pero debido a la elevada tasa de error del reconocedor sólo funcionaron para algunos de los ficheros y se obtenía un número muy

limitado de etiquetas.

Por último, se decidió determinar de forma manual los tiempos equivalentes en las transcripciones del reconocedor, obteniendo así las 554 etiquetas temporales, de forma que se puedan evaluar respecto a las de referencia.

Con el objetivo de mostrar el rendimiento del reconocedor y analizar la posibilidad de probar este método de alineamiento con un modelo más avanzado de cara a un futuro, se calculó la tasa de *Word Error Rate* (**WER**) para las palabras final de frase de cada fichero (que es donde se encuentran las etiquetas).

$$\mathtt{WER} = \frac{\mathtt{S} + \mathtt{B} + \mathtt{I}}{\mathtt{N}}$$

Figura 3.6: Word Error Rate

En la ecuación de la figura anterior se muestra la expresión que permite realizar el cálculo del WER, donde S es el número de sustituciones, B el de borrados, I el de inserciones y N el número total de palabras de final de frase en el fichero de texto de referencia.

3.4. Evaluación de los Alineadores

Una vez se tuvieron todos los intervalos de tiempo de cada uno de los segmentos en los que se dividen los textos para cada uno de los alineadores, se recorrieron las etiquetas una a una y se calculó la diferencia en ms entre el inicio y el final de cada una, obteniendo así los tiempos de error para cada etiqueta de cada fichero.

Se obtuvo un valor de tiempo de error por cada límite entre alineamientos de frase, que permitió calcular otros valores de error a nivel de fichero.

Ya que, dentro de los 20 ficheros de audio el número de frases es diferente, se decidió tomar como medida de error la raíz del error cuadrático medio o *Root-Mean Square Error* (**RMSE**), ya que tiene en cuenta el número de frases o etiquetas de cada uno. Éste suele utilizarse como medidor de las diferencias entre las predicciones de un modelo, es siempre mayor que 0, y cuanto menor sea su valor mayor será la predicción del modelo. Tomando X_i y Y_i como el instante en segundos de las etiquetas de referencia y del alineador para cada frase de índice **i**, y **n** el número de frases de cada fichero, el RMSE de cada fichero viene dado por la siguiente expresión.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(X_i - Y_i\Big)^2}$$

Figura 3.7: Root-Mean Square Error

Además, se observó que los alineamientos generados por los alineadores tienden a marcar las etiquetas temporales en el instante inmediato al terminar cada frase o en el inicio de la siguiente, mientras que en las etiquetas generadas manualmente se marcaron en el punto medio del silencio entre frases. Ya que a priori el que no coincidan en estos puntos no es un indicador de que el alineador no tenga un buen rendimiento, se decidió calcular aquellos valores de error que se correspondían a intervalos de silencio de la señal.

Para su cálculo se utilizó un detector de actividad de voz o *Voice Activity Detector* (**VAD**) implementado por el grupo **AUDIAS** para obtener aquellos intervalos temporales de cada fichero

de voz en los que se encuentra la señal de voz, y aquellos en los que hay un silencio. Una vez se obtienen para cada uno de los textos, se implementó una función de Matlab para procesar aquellos intervalos en los que hay silencio, y posteriormente determinar qué diferencias entre etiquetas o valores de error se deben a que se encuentran en diferentes puntos de dichos intervalos de silencio. El valor de error utilizado se obtuvo de la siguiente forma. En primer lugar se calcularon las diferencias en segundos entre las etiquetas de referencia y las etiquetas generadas por cada alineador, obteniendo la suma total para cada fichero. Después se analizaron aquellos fragmentos de las diferencias temporales que se encontraban en partes de silencio de la señal, y se sumaron para cada fichero. Por último se dividió el error correspondiente a la parte de silencio de la señal entre el error total calculado en la primera parte, para cada fichero, obteniendo un valor de porcentaje de error que se debido a silencio, tal y como se muestra en la siguiente figura.

$$Error = \frac{\Sigma_{i=1}^{n} |Error_{sil}|}{\Sigma_{i=1}^{n} |Error_{tot}|} \texttt{ * 100}$$

Figura 3.8: Porcentaje de error debido a silencio

Este porcentaje de error sirve para ver el error de cada fichero que no se debe tener en cuenta, ya que el instante dentro del silencio entre dos frases en donde se marque el fin de frase no tiene relevancia a la hora de realizar los alineamientos, siempre que se encuentre dentro de éste.

El último experimento que se incluirá será el del cálculo de la distribución de los valores de las diferencias entre etiquetas. Esto se hizo con el objetivo de ver, no sólo los valores de tiempo en torno a los que se desvía cada alineador, sino también si tienden a adelantarse o atrasarse temporalmente respecto a las etiquetas de referencia.

Capítulo 4

Resultados

En este capítulo se describen los experimentos realizados para cada uno de los alineadores forzados de la evaluación. El objetivo de esta sección es el de evaluar a partir de los resultados obtenidos para cada alineador forzado cuál de ellos es más acorde de cara a utilizarlo en la generación de una base de datos que sirva posteriormente para entrenar un reconocedor de voz en español. Para ello, se muestran diferentes gráficas indicativas del rendimiento de cada uno, lo que sumado a otros aspectos a tener en cuenta, como la licencia, la facilidad de uso, o el tiempo de ejecución serán de ayuda para determinar cuál es más acorde al problema del que trata este Trabajo de Fin de Máster.

4.1. Aeneas

Aeneas es un alineador forzado que, como se describe en 2.7.6, es muy sencillo de utilizar. Para obtener los datos de alineamiento se implementó en Python un sistema automático mediante el cual se pueden obtener todos los alineamientos de audio y texto a partir de un directorio dado en el que se encuentren los pares de ficheros de audio y texto.

Una vez que se obtuvieron los alineamientos, se procesaron y obtuvieron valores de error, tal y como se describe en 3.2.1.

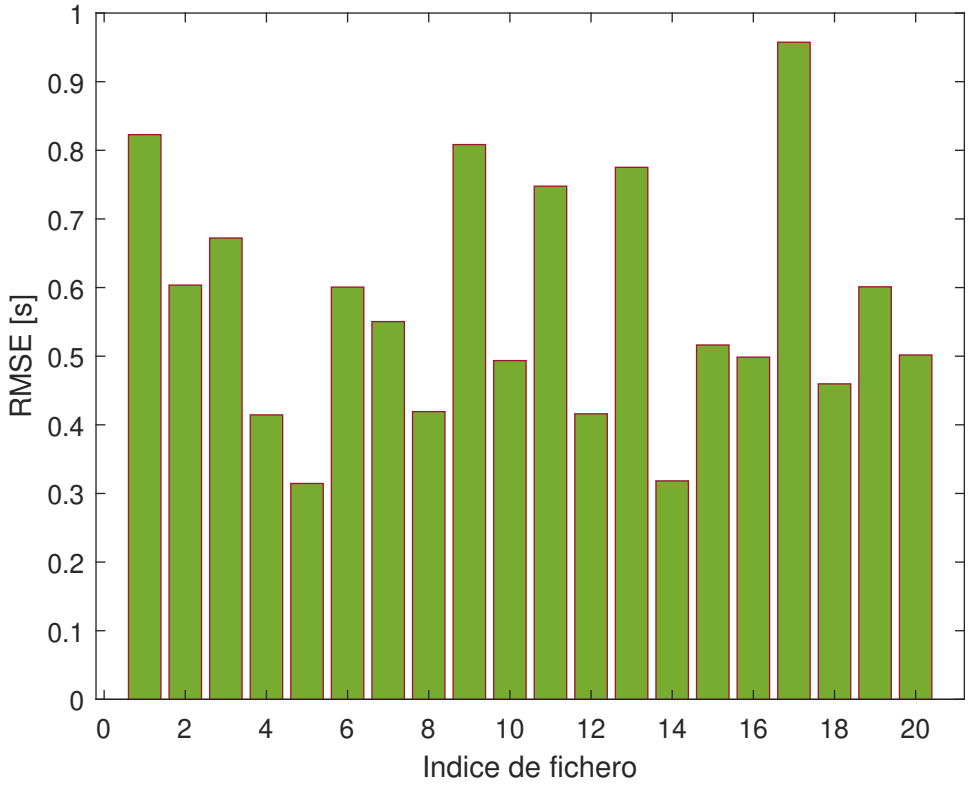

Figura 4.1: Raíz del error cuadrático medio en Aeneas

En la figura 4.1 se muestran los diferentes valores de RMSE para cada uno de los ficheros evaluados. Este error es el indicativo principal del rendimiento de cada alineador, siendo 0 el valor óptimo y peor el rendimiento cuanto mayor es el valor. Los valores se encuentran entre 0.3 y 0.95 segundos y, como se verá más adelante, se reducirían considerablemente si no se tuviera en cuenta el error debido a silencio, que se mostrará en la figura 4.2.

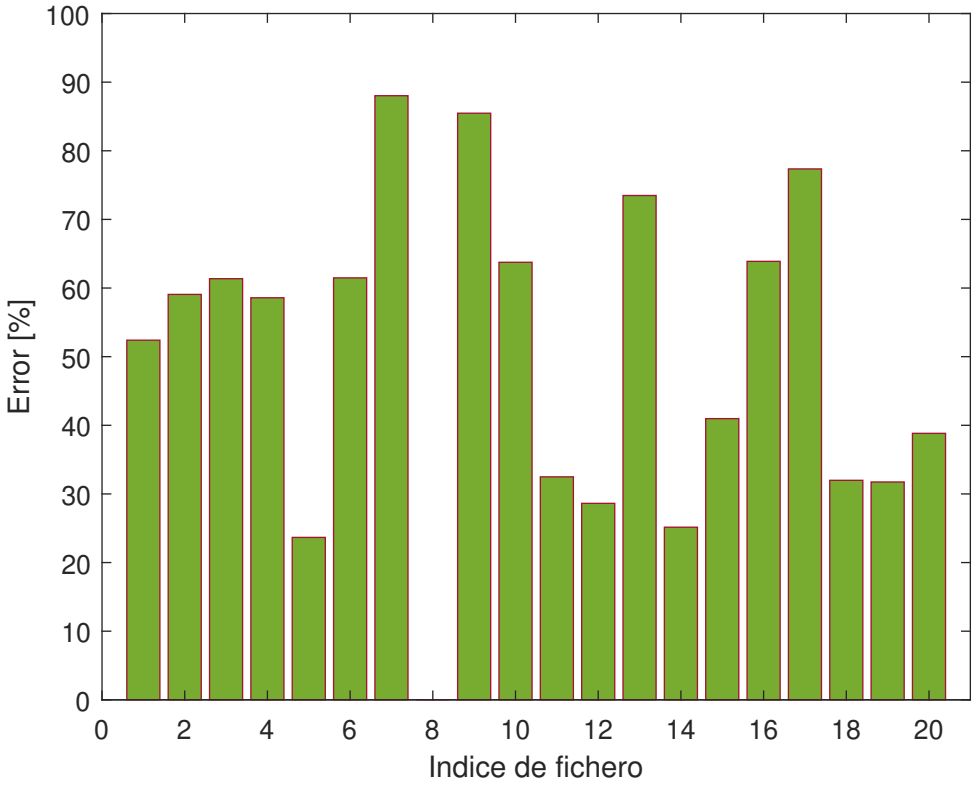

Figura 4.2: Error por silencio entre frases en Aeneas

En la figura 4.2 se muestra el porcentaje de error que se encuentra en un intervalo de silencio.

Los valores de error son muy elevados en la mayoría de los ficheros, estando por encima del 50 % de error en 11 de los 20, y los restantes por encima del 25 %.

En el fichero número 8 el porcentaje de error es nulo. Esto es debido a que el detector de actividad utilizado no obtiene resultados muy precisos cuando se trata de segmentar ficheros de audio con ruido, o cuyas pausas son muy pequeñas. Cuando los silencios que hace el lector son demasiado cortos, el VAD no los detecta y toma al fichero como una única o muy pocas frases. Tras analizar manualmente los archivos de salida del detector de actividad, se observa que algunos de los ficheros tenían definidos segmentos de voz mucho más largos y un número de silencios menor. Después de escuchar todos los ficheros de audio se observó que aquellos en los que se hacían menos pausas eran aquellos en los que el detector no realizó la división en voz y silencio para todas las pausas que se hacían. Por tanto, de cara a líneas futuras de trabajo, es interesante implementar un detector de actividad de voz más agresivo, que, aunque divida en exceso el fichero en demasiadas frases, obtenga todos los intervalos de silencio, de cara a evaluar el valor real de error debido a silencio de cada alineador.

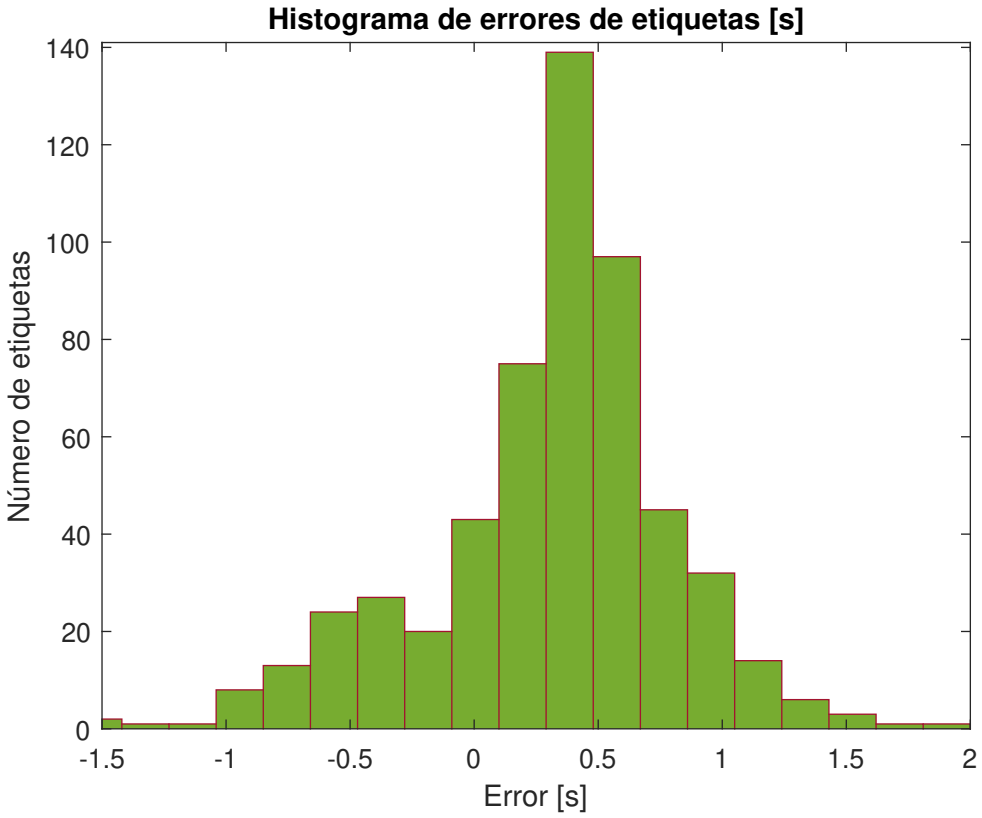

Figura 4.3: Distribución de los valores de error en Aeneas

En la figura 4.3 se puede ver la distribución de los valores de tiempo en segundos en los que se diferencian las etiquetas de Aeneas respecto de las etiquetas de referencia. Los valores siguen una distribución similar a una gaussiana, oscilando entre -2 y 2 segundos. Los valores negativos se deben a aquellos errores que se deben a que el alineamiento generado se ha dado en un instante posterior al del alineamiento manual, y los valores positivos a aquellas etiquetas que se han generado en instantes anteriores a los de las etiquetas de referencia. Por ello se puede determinar que el alineador tiende a poner las etiquetas en instantes anteriores a las etiquetas generadas manualmente.

Los valores de error más comunes están entre 0 y 0.5 sin contar que, gran parte de esos errores son debidos a la parte del silencio donde se realizó el alineamiento. El error máximo se encuentra en torno a 2 segundos, en un porcentaje muy bajo de las etiquetas. Esto es algo muy a tener en cuenta, ya que los errores grandes que sobrepasen la duración de un silencio pueden dar lugar a cortar segmentos de audio que no correspondan en tareas de segmentación de voz necesarias para preparar los datos de entrada en los sistemas de ASR.

4.2. Montreal Forced Aligner

Para obtener los resultados del MFA se optó por realizar los alineamientos a partir del modelo acústico pre-entrenado en español que se puede encontrar en su web de referencia, ya que está entrenado con un número de horas mucho mayor que el de la base de datos que se implementó. De hecho, se realizaron pruebas mediante el entrenamiento de un modelo acústico para cada fichero y locutor, y los resultados obtenidos fueron peores que utilizando el modelo pre-entrenado. Como excepción, el fichero para el que peores resultados se obtuvieron, el número 16, dio problemas al alinearse mediante el modelo acústico pre-entrenado, y se entrenó un modelo acústico específico para alinear este audio.

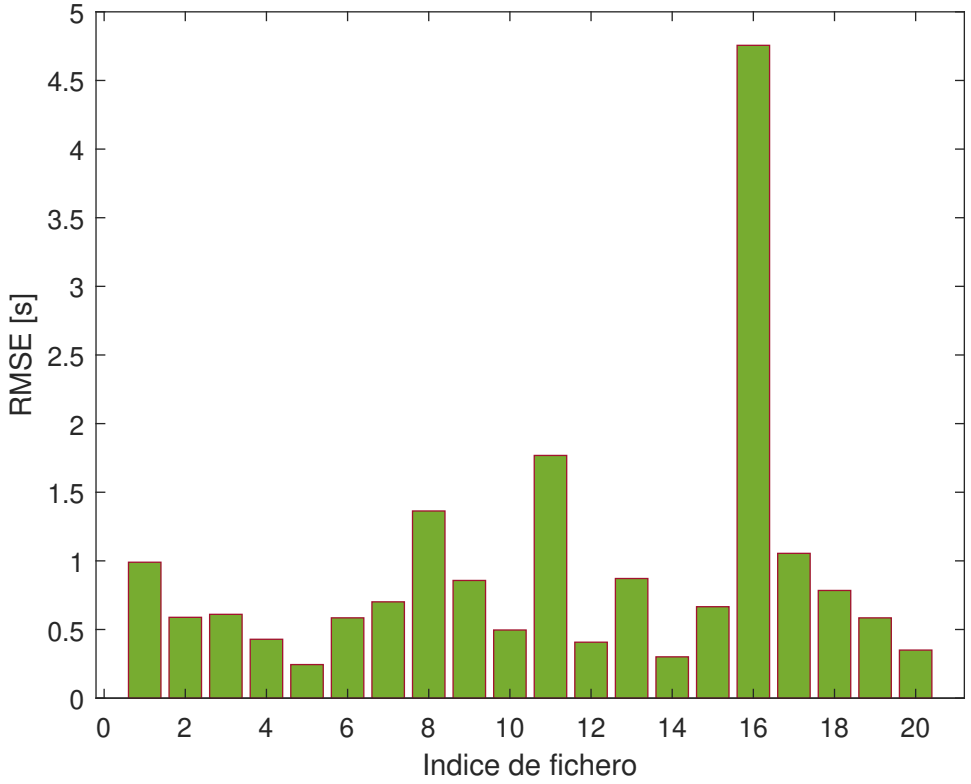

Figura 4.4: Valores de RMSE en el MFA

En la gráfica de 4.4 se pueden observar los valores de RMSE para el MFA. La mayor parte de los ficheros tienen valores de error por debajo de 1 segundo, similares a los obtenidos en Aeneas. Sin embargo, en algunos de los ficheros el RMSE está por encima de 1, y en concreto el número 16, que llega a un error cercano a los 5 segundos. Cabe destacar que se esperaba un rendimiento en el MFA mucho mayor al del resto de alineadores, ya que en inglés es el más avanzado y utilizado en la actualidad. Este modelo, tal y como se especifica en su documentación está diseñado para funcionar de forma óptima con ficheros de 30 segundos, mientras que los de las pruebas realizadas tienen una duración en torno a 300 segundos. Además, en el procesado de los datos de alineamiento se vio que tenía dificultades alineando algunos caracteres y fonemas propios del español, y es por ello que se suprimieron aquellas etiquetas correspondientes a los finales de frase en los que se encontraban estas palabras. Esto no dificultó la evaluación, pero dado que el número de etiquetas en el cálculo del RMSE es menor, el valor de éste tiende a subir en algunos de los ficheros con menos etiquetas.

Por todo ello se puede decir que el MFA tiene peor rendimiento que Aeneas, a pesar de tener un buen rendimiento para la mayoría de los ficheros.

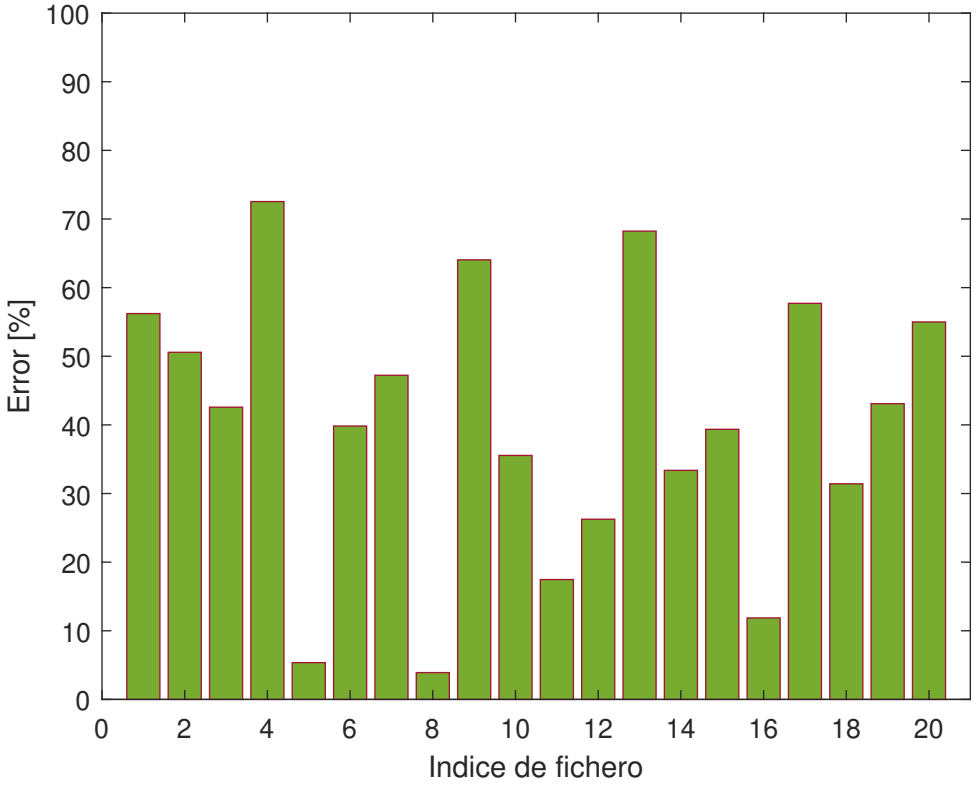

Figura 4.5: Error debido a silencio en el MFA

En la figura 4.5 se pueden ver los valores de error causados por el silencio entre frases de forma análoga a como se hizo para Aeneas. En líneas generales los valores de error son menores que en el Aeneas, aunque siguen siendo valores muy grandes, por encima del 30 % en 15 de los 20 ficheros. Se puede ver que el fichero 8 sigue teniendo un valor casi nulo al igual que ocurría en Aeneas, ya que, como se explicó, viene dado por el bajo rendimiento del detector de actividad de voz. También destaca el número 5, que como se observa en 4.4 es el que menor valor de error tiene, y por tanto tiene sentido que el error por silencio sea menor en este caso.

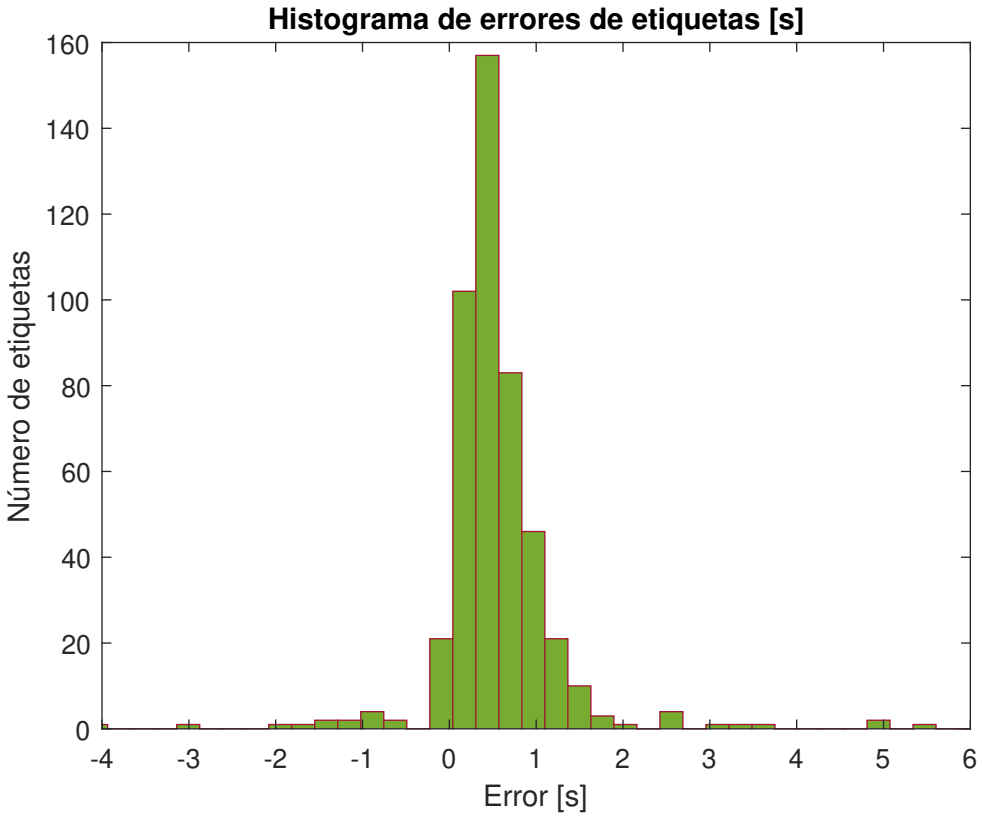

Figura 4.6: Distribución de los valores de error en MFA

Por último, se muestra el histograma con los valores acumulados de error para este alineador en la figura 4.6. La distribución de los valores de error es similar a la obtenida en Aeneas, con una distribución casi gaussiana, centrada en torno a 0.3 segundos y cuyos valores de error oscilan entre -2 y 2.5. Se puede ver que, al igual que ocurre en Aeneas, la tendencia del alineador es a adelantarse y poner las etiquetas en los finales de frase, y no en el punto medio del silencio entre frases, tal y como se hizo al crear las etiquetas de referencia.

Destaca que la mayoría de las etiquetas tienen un error comprendido entre 0 y 1, lo que supone un buen rendimiento en los alineamientos. Como aspecto negativo se pueden ver outliers en valores de error mucho mayores, que llegan hasta los 5.5 segundos. Estos valores se deben al alineamiento del fichero 16, para el que el rendimiento del MFA fue mucho menor al del resto.

4.3. Resultados del alineador forzado propio

En esta sección se describirán los resultados obtenidos para el alineador propio que se mostró en 3.3, de forma análoga a como se hizo en los dos alineadores anteriores. Además, se medirá el rendimiento del reconocedor de voz utilizado en los finales de frase, con el objetivo de verificar que, para un modelo más avanzado y entrenado con más horas, este sistema tendría un rendimiento mucho mayor.

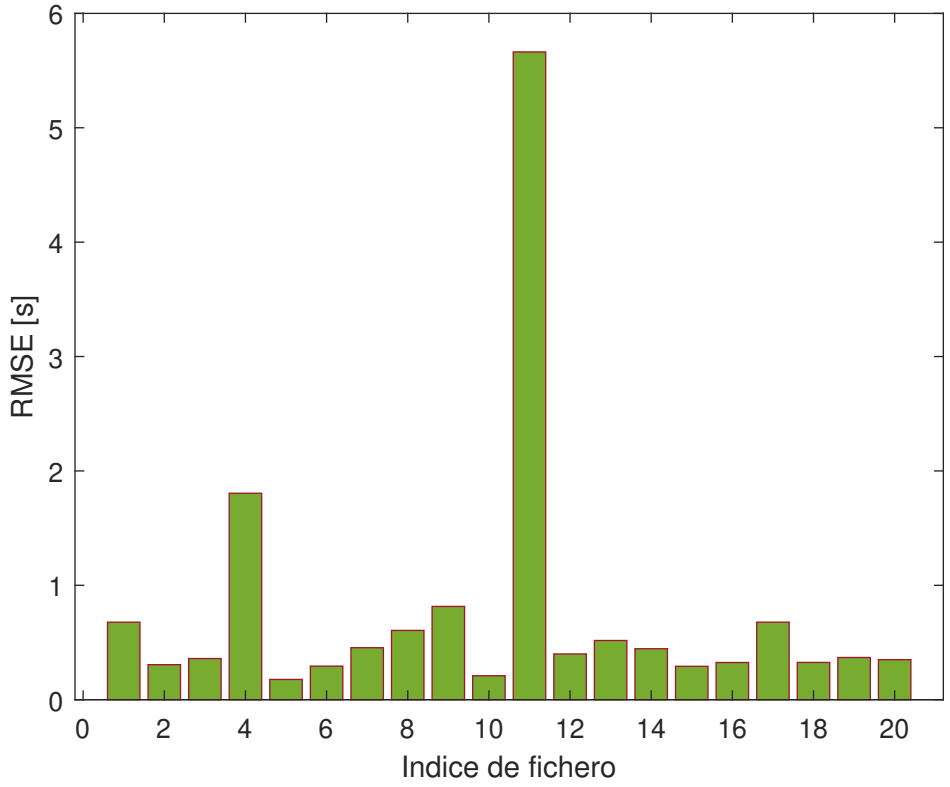

Figura 4.7: Valores de RMSE en el Alineador Propio

En la figura 4.7 se muestran los distintos valores de RMSE para cada uno de los ficheros evaluados. El rendimiento del reconocedor de voz situando las palabras en los instantes en los que se han pronunciado es muy bueno, a excepción de el fichero 11, para el que la transcripción, tras revisarla manualmente, se vio que tenía un porcentaje de acierto en las palabras muy bajo. Los valores de error se encuentran por debajo de 1 en 18 de los 20 ficheros, y por debajo de 0.5 segundos en 12 de ellos. La limitación principal de este modelo es la siguiente. Si el reconocedor obtiene la transcripción de forma aproximada, aunque no exacta, el alineamiento llega a ser bastante bueno. En el caso de que no consiga reconocer una parte del audio o se salte algunas frases durante el reconocimiento, como ocurre en el fichero 11, hay etiquetas cuyo error es muy elevado y el rendimiento es muy inferior.

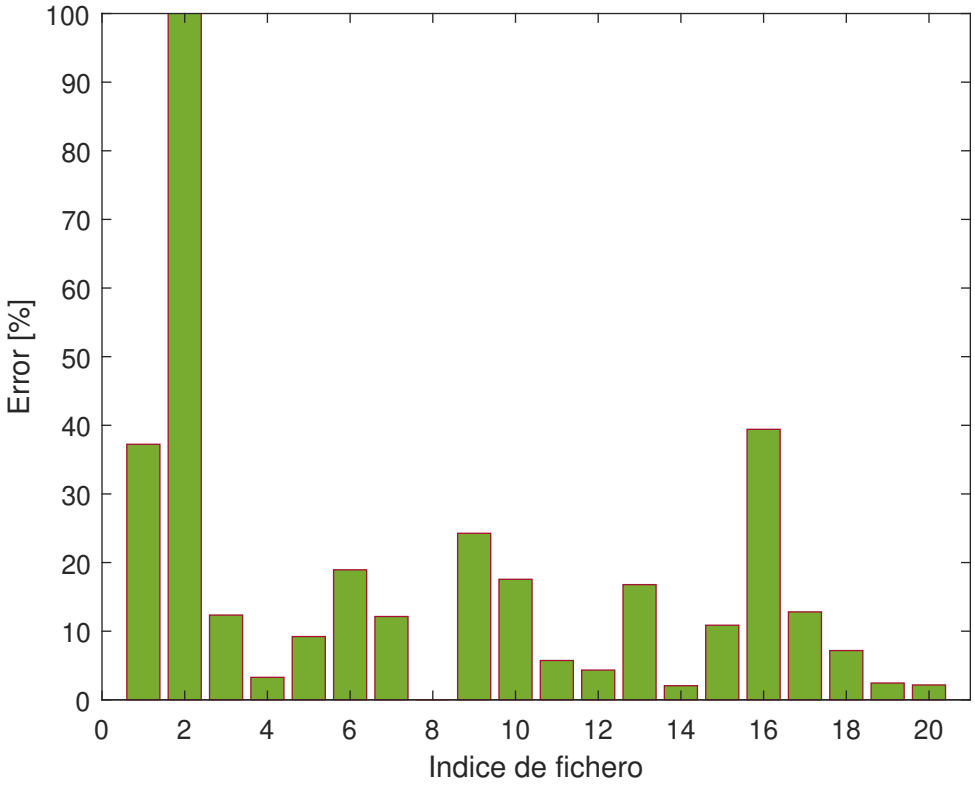

Figura 4.8: Error debido a silencio en el Alineador Propio

En cuanto al porcentaje de error que se debe a silencio en este alineador, mostrado en la figura 4.8, se puede ver cómo en este modelo los valores de error de este tipo son mucho menores que en los anteriores. Todos los ficheros menos uno se encuentran por debajo del 40 % de error, y 8 de ellos por debajo del 10 %. Destaca el fichero número 2, para el que el 100 % de los errores cometidos se encuentran en intervalos de silencio, por lo que su rendimiento en este caso sería excelente. El resto de valores, es normal que puedan ser más bajos debido al acierto del detector de actividad en algunos de los ficheros, pero no llegan a serlo en el Aeneas o el MFA.

Por tanto, se puede afirmar que la cantidad de error que no es real en este alineador es mucho menor que en el resto, y, por tanto, su rendimiento, que parece a priori mejor que el de los dos modelos anteriores en la mayoría de ficheros, es menor de lo esperado.

En la distribución de los valores de error mediante el histograma de 4.9 se observa que la mayoría de los errores se encuentran entre 0 y 0.25 segundos, por lo que su rendimiento global es muy bueno, mejor que el de los dos alineadores anteriores, a pesar de que algunos ficheros no se hayan alineado con esa precisión, como se vio en la figura que muestra el error RMSE en 4.7.

No hay apenas etiquetas cuyo error sea superior a 1 segundo, y los errores que sobrepasan este límite se corresponden al fichero para el cuál la transcripción de texto obtenida por el reconocedor no fue buena. En la siguiente sección, se mostrará una tabla comparativa entre los porcentajes de los valores absolutos de error entre los tres alineadores, para ver el peso que realmente tienen los valores grandes de error y determinar finalmente cuál de ellos rinde mejor de cara al objetivo final del trabajo.

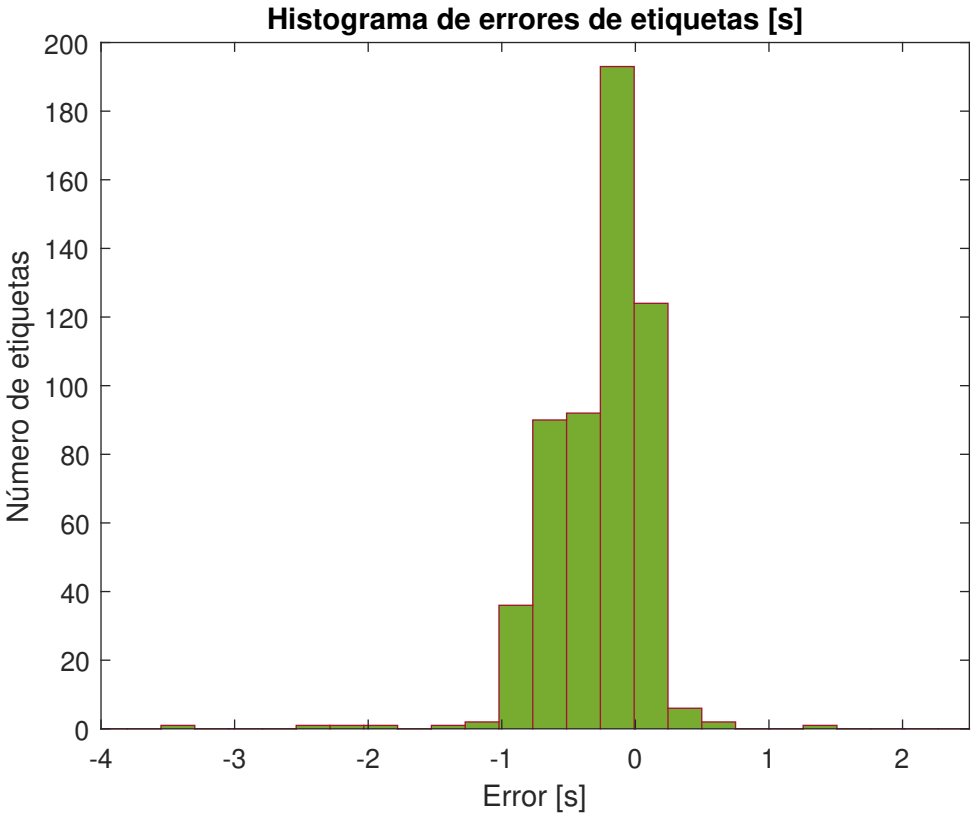

Figura 4.9: Distribución de los valores de error en el Alineador Propio

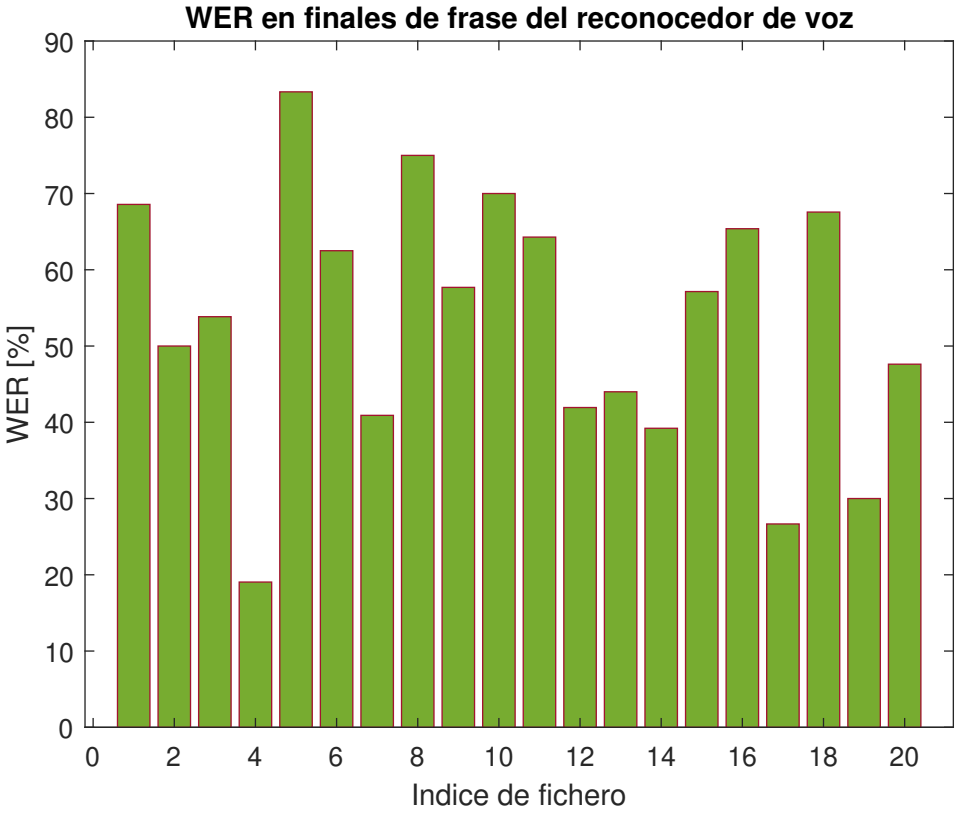

Figura 4.10: WER en las palabras de final de frase reconocidas por el Alineador Propio

En la figura 4.10 se muestra un valor de WER para cada fichero de texto. Este porcentaje de error se calculó a partir de los aciertos en las palabras reconocidas en los finales de frase de cada fichero de texto (punto donde se marcan las etiquetas) respecto del número total de palabras de final de frase en las transcripciones del reconocedor de voz utilizado para implementar el alineador propio.

Se puede ver que, exceptuando ficheros como el 4, el 17, o el 19, los niveles de WER son muy elevados en los finales de frase, por valores por encima del 40 %. El reconocedor utilizado alinea con un gran rendimiento los segmentos de audio y texto cuando consigue reconocer las palabras de forma precisa, sin embargo esto no ocurre en casi ninguno de los ficheros, y es por eso que los valores de error obtenidos dan una aproximación, pero no el error real que tiene este sistema como alineador forzado. De cara a futuros trabajos conviene implementar este tipo de alineador utilizando un reconocedor de voz más moderno, entrenado con una cantidad de datos mayor, y cuya fase de alineamiento de textos, en lugar de hacerse por separado, vaya integrada dentro del mismo.

4.4. Comparación entre alineadores

En esta sección se hará una comparación entre los tres alineadores en base a los resultados obtenidos anteriormente, y se evaluará cuál de todos es más adecuado de cara a su utilización en el Trabajo de Fin de Máster complementario.

En la siguiente tabla se permite ver el rendimiento de cada uno de los alineadores para distintos valores de tolerancia (porcentaje por debajo de un límite) en milisegundos.

	Tolerancia [ms]			
	< 250 ms	< 500 ms	< 750 ms	< 1000 ms
Aeneas	0.215	0.587	0.820	0.926
MFA	0.183	0.506	0.717	0.845
Propio	0.561	0.738	0.906	0.978

Tabla 4.1: Precisión a diferentes tolerancias en los tres alineadores

La comparación propuesta en la tabla 4.1 muestra los valores de precisión correspondientes a diferentes valores de diferencia entre etiquetas sintéticas y de referencia que se encuentran por debajo de determinados umbrales establecidos (tolerancias).

El alineador que más precisión tiene es el propio, creado a partir de un reconocedor de voz. Éste tiene un 56.1 % de etiquetas cuyo error está por debajo de los 250 ms, con un rendimiento notablemente superior al sus competidores. A partir de los 250 ms el rendimiento de los tres se iguala, teniendo una diferencia de 13.3 puntos en el rendimiento por debajo de 1000 ms entre el MFA y el alineador propio, y de un 5.2 entre éste último y Aeneas.

	Media (ms)	Mediana (ms)
Aeneas	490	438
MFA	692	497
Propio	405	205

Tabla 4.2: Media y mediana del error entre etiquetas (ms)

En la tabla de 4.2 se puede ver la comparación entre las medias y las medianas para las diferencias entre etiquetas en los tres alineadores forzados de la evaluación.

En cuanto a la media, la más baja es la del alineador propio, seguida de la de Aeneas, y claramente por debajo de la del MFA, con una media de casi 0.7 segundos.

En el caso de las medianas la diferencia entre el alineador propio y el resto es aún mayor, mientras que la de Aeneas y el MFA es prácticamente igual.

Por tanto, se puede confirmar que, a pesar de no estar diseñados para tareas como la que se propone, con segmentos de audio muy largos, los tres alineadores cumplen su función de forma más o menos precisa. Los valores de error de los tres modelos oscilan en su mayor parte entre 0 y 1 segundos, valores que no exceden los límites necesarios, ya que la tarea para la que se utilizarán es la de segmentar textos muy grandes para la creación de bases de datos.

En cuanto a su rendimiento, el que mejor resultados ofrece es el alineador propio. Este modelo determina con gran precisión en qué instante empieza y termina cada frase, sin embargo, su imprecisión a la hora de reconocer las palabras no sólo de forma exacta, sino aproximada, dificultan la automatización de su proceso de alineamiento, y fue para el cuál llevó un mayor número de horas obtener los resultados.

Los resultados del MFA son buenos teniendo en cuenta que los modelos acústicos utilizados no se habían entrenado para los datos que se estaban alineando, además de haberse entrenado en español latino. Este modelo es el que mejor resultado da en frases muy cortas y a nivel de fonema y palabra, pero a medida que se utilizan frases cada vez más largas disminuye su rendimiento.

Por último, el Aeneas es más sencillo de utilizar que los dos anteriores, ya que no requiere entrenamiento del modelo de lenguaje y acústico, ni una generación del léxico. Sus resultados mejoran a los del MFA, y su ejecución tarda en torno a 30 veces menos en las pruebas realizadas. Su rendimiento a nivel de palabra y fonema es nulo, ya que aporta alineamientos únicamente a nivel de frase, pero dado que el problema que se plantea solo requiere este tipo de alineamiento, es el alineador más adecuado, y será el que se utilizará en el segundo Trabajo de Fin de Máster en [1].

Capítulo 5

Conclusiones y trabajo futuro

Los sistemas de alineamiento forzado de audio y texto son muy necesarios en la actualidad. Además, la variedad entre aquellos cuya licencia permite su libre utilización es muy escasa. Por ello, se analizaron la mayoría de estas herramientas con el objetivo de determinar cuál de ellas se adecuaba más a la construcción de una base de datos en español enfocada al entrenamiento de sistemas de reconocimiento de voz. Tras hacer un estudio de la mayoría de ellos se optó por implementar un sistema automático de evaluación para dos de los modelos: el Alineador Forzado de Montreal y el Aeneas. Además, se añadió a la evaluación un alineador original, basado en el uso de un reconocedor de voz del grupo AUDIAS. Para realizar la evaluación se optó por utilizar una base de datos comprobada manualmente, de forma que no hubiera ningún error en las medidas de rendimiento de los alineadores. Para ello se elaboró manualmente una base de datos de 20 ficheros de audio y texto, sobre la que se determinarían alineamientos de referencia, también de forma manual, para así tener unos datos sobre los que realizar la evaluación. Tras procesar sus datos se obtuvieron unos resultados que sirvieron para sacar una serie de conclusiones que se expondrán a continuación.

En primer lugar, los tres alineadores forzados tienen un rendimiento adecuado al problema que se plantea, ya que, para alinear grandes ficheros de audio no es necesario tener unos alineamientos con un nivel de error excesivamente bajo. Esto se debe a que la finalidad es segmentar ficheros de larga duración en otros de menor tamaño, y si el error de alineamiento no sobrepasa los 2 segundos puede servir como solución al problema planteado.

En cuanto al Alineador Forzado de Montreal, es el que ofrece mayor precisión a nivel de palabra y frase, para segmentos de duración inferior a 30 segundos. Sin embargo, el utilizarlo en ficheros demasiado grandes hace que sus niveles de error aumenten llegando a tener problemas en algunos de los ficheros de voz. Además, la necesidad de utilizar un modelo de lenguaje, un modelo acústico y un léxico hace que su uso sea más complejo que el de los otros alineadores de la evaluación. Dentro de las lineas de trabajo futuro entraría el entrenar un modelo acústico en español con una librería de un número de horas mayor, para evaluar si su rendimiento aumenta de cara a utilizarlo en ficheros de audio de mayor tamaño.

El alineador propio ofrece resultados muy buenos, con niveles de error menores a los del MFA y Aeneas. Sin embargo, el elevado nivel de WER en sus transcripciones imposibilitaron la automatización del proceso de alineamiento, teniendo que hacerlo manualmente para obtener todas las etiquetas temporales correspondientes. Por ello es interesante probar el mismo sistema de alineamiento para un reconocedor similar más avanzado y entrenado con un número de horas mayor, para ver si sirve como solución en la elaboración de bases de datos.

Aeneas es el modelo más equilibrado, ofreciendo buenos resultados, con un error bajo y debido en gran parte al silencio entre frases, por lo que su error real sería aún mucho menor.

Su uso es muy cómodo y permite integrarlo en cualquier sistema de forma sencilla, puesto que se encuentra en Python, y su velocidad de ejecución es la más rápida con respecto al resto. Por todas estas cosas este será el alineador que se utilizará en el segundo TFM para elaborar la base de datos en español.

Como posible trabajo futuro entraría también el de entrenar un sistema de detección de actividad de voz (VAD) mucho más agresivo a la hora de detectar los silencios, y establecer de forma precisa qué porcentaje de error de los alineadores realmente no tiene importancia y poder marcar como error sólo aquellas etiquetas que se excedan del intervalo que divide ambas frases.

Bibliografía

[1] Luis Miguel Martínez Antolín. Generación de una base de datos para reconocimiento de voz en español mediante alineamiento de audio y texto. 2020.

[2] John Levis and Ruslan Suvorov. *Automatic Speech Recognition*. 11 2012.

[3] R. Schmidt and R. Neumann. Automatic text-to-speech alignment: Aspects of robustification. In Václav Matousek, Pavel Mautner, Jana Ocelíková, and Petr Sojka, editors, *Text, Speech and Dialogue*, pages 72–76, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[4] Omer Gold and Micha Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *ACM Trans. Algorithms*, 14(4), August 2018.

[5] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[6] Biing Hwang Juang and Laurence R Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[7] Steve J Young and Sj Young. *The HTK hidden Markov model toolkit: Design and philosophy*. University of Cambridge, Department of Engineering Cambridge, England, 1993.

[8] Roberto Carrillo Aguilar. Diseño y manipulación de modelos ocultos de markov, utilizando herramientas htk. una tutoría. *Ingeniare. Revista chilena de ingeniería*, 15(1):18–26, 2007.

[9] David Rybach, Stefan Hahn, Patrick Lehnen, David Nolden, Martin Sundermeyer, Zoltan Tüske, Simon Wiesler, Ralf Schlüter, and Hermann Ney. Rasr-the rwth aachen university open source speech recognition toolkit. In *Proc. ieee automatic speech recognition and understanding workshop*, 2011.

[10] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Vesel. The kaldi speech recognition toolkit. *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 01 2011.

[11] Simon Gonzalez, James Grama, and Catherine E Travis. Comparing the performance of major forced aligners used in sociophonetic research.

[12] Paul Lamere, Philip Kwok, Evandro Gouvea, Bhiksha Raj, Rita Singh, William Walker, Manfred Warmuth, and Peter Wolf. The cmu sphinx-4 speech recognition system. 1:2–5, 2003.

[13] Robert Ochshorn and Max Hawkins. Gentle forced aligner. *URL: http://lowerquality.com/gentle/(visited on 09/27/2019)*, 2016.

[14] Sravana Reddy and James N Stanford. Toward completely automated vowel extraction: Introducing darla. *Linguistics Vanguard*, 1(1):15–28, 2015.

[15] Jiahong Yuan and Mark Liberman. Speaker identification on the scotus corpus. *Journal of the Acoustical Society of America*, 123(5):3878, 2008.

[16] Akinobu Lee and Tatsuya Kawahara. Recent development of open-source speech recognition engine julius. pages 131–137, 2009.

[17] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *INTERSPEECH*, 2017.

[18] Athanasios Katsamanis, Matthew Black, Panayiotis G Georgiou, Louis Goldstein, and S Narayanan. Sailalign: Robust long speech-text alignment.

[19] E. Wilbanks. fasealign (version 1.1.9) [computer software]. retrieved oct 11, 2018 from https://github.com/ericwilbanks/fasealign. 2008.

[20] Jean-Philippe Goldman and Sandra Schwab. Easyalign spanish: An (semi-)automatic segmentation tool under praat. 01 2014.

[21] Mart Lubbers and Francisco Torreira. Praatalign: an interactive praat plug-in for performing phonetic forced alignment. `https://github.com/dopefishh/praatalign`, 2013-2018. Version 2.0.

[22] Robert Fromont. Forced alignment of different language varieties using labb-cat.

[23] Francesco Cangemi, Francesco Cutugno, Bogdan Ludusan, Dino Seppi, and Dirk Van Compernolle. Automatic speech segmentation for italian: tools, models, evaluation, and applications.

[24] Brigitte Bigi and Daniel Hirst. Speech phonetization alignment and syllabification (sppas): a tool for the automatic analysis of speech prosody. 05 2012.

[25] Aeneas alignment tool. `http://www.readbeyond.it/aeneas/docs/`. Accessed: 2019-09-30.

[26] Jodi Kearns. Librivox: Free public domain audiobooks. *Reference Reviews*, 2014.

[27] Project gutenberg. (n.d.). retrieved february 21, 2016, from www.gutenberg.org. 2020.

[28] Tomas Boril and Radek Skarnitzl. Tools rpraat and mpraat. volume 9924, pages 367–374, 09 2016.